AN ALGEBRAIC APPROACH TO COMPUTING INVERSE LAPLACE TRANSFORMS OF RATIONAL FUNCTIONS

by

JAMES FLOYD STAFFORD

B.S. Kansas State University, 1986

A MASTER'S THESIS
submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1989

Approved by:

Major Professor

LD
2668
.T4
EECE
1989
S73
c.2

A11208 618206

Table of Contents

List of Tables

List of Figures

Chapter I

Introduction

The electrical engineer must often solve linear, constant-coefficient, differential equations, especially in such fields as circuit analysis and control theory. The usual method of solution involves use of the Laplace transform. Although this method of solution is well-known [1], computing solutions to such equations often involves many tedious calculations. The most difficult step in this method of solution is performing the inverse Laplace transform of a rational function. The object of this thesis is to describe an algorithm for solving large problems of this kind.

Solving such equations is not necessarily a numerical analysis problem. We are not trying to approximate the solution to a differential equation. We already know the general form of the solution, and are simply seeking coefficients for the solution. Thus the problem is actually algebraic, rather than analytic.

There are three distinct steps in computing the inverse Laplace transform of a rational function. They are: factoring the denominator polynomial, finding the partial fraction expansion of the rational function, and computing the inverse Laplace transform of each of these partial fractions. Except for the factoring, these are not analytic problems, and even the factoring algorithm presented here makes use of an algebraic technique to speed up the finding of multiple roots.

Special concern was devoted to the partial fraction expansion portion of the algorithm in order to reduce the number of calculations. The method presented herein is based on one developed elsewhere [4] and modified to further reduce the number of calculations. The effort to minimize the number of calculations required constitutes the main contribution of this research.

As the title declares, this thesis concerns itself with computing the inverse Laplace transform of a rational function, but not with obtaining the rational function in the first place. The algorithm consists of four parts. The first part of the algorithm accepts a rational function as input and outputs the rational function with the denominator expressed as a product of irreducible factors. The next part accepts as input a rational function with a factored denominator. It outputs the partial fraction expansion of the given rational function. The third stage accepts this expansion and computes the inverse Laplace transform. The final stage evaluates the inverse transform over a desired interval and plots a graph, if desired.

Chapter II

Some Preliminaries

We first review the definition of the Laplace transform and its usefulness in solving linear, constant-coefficient, differential equations. Given a real-valued function $f(t)$ of a real variable, its one-sided Laplace transform, $F(s)$, is given by [1]:

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^{+\infty} f(t)e^{-st}dt,$$

where s is complex and hence $F(s)$ is, in general complex-valued. This transform has two important properties which can be used to transform a differential equation in t into an algebraic equation in s. Both of these properties are easy to derive and proofs are given elsewhere. These properties are [1]:

Linearity. Suppose $f_1(t)$ and $f_2(t)$ are such that their Laplace transforms, $F_1(s)$ and $F_2(s)$, exist. Also, let $c_1, c_2 \in \mathbf{R}$. Then

$$\mathcal{L}\{c_1 f_1(t) + c_2 f_2(t)\} = c_1 F_1(s) + c_2 F_2(s).$$

Forward Derivative Property. Suppose that $f(t)$ is $(n-1)$ times continuously differentiable and that the nth derivative, $f^{(n)}(t)$, is such that its Laplace transform exists. Then

$$\mathcal{L}\{f^{(n)}(t)\} = s^n F(s) - \sum_{j=1}^{n} s^{n-j} f^{(j-1)}(0).$$

Given an nth-order linear differential equation with constant coefficients

$$\sum_{j=0}^{n} a_j f^{(j)}(t) = b(t),$$

and the initial conditions, $f(0) = c_0, f^{(1)}(0) = c_1, \ldots, f^{(n-1)}(0) = c_{n-1}$, we can take the Laplace transform of both sides of the equation and end up with [1]

$$F(s) = \frac{B(s) + \sum_{i=1}^{n} c_{i-1} \sum_{j=i}^{n} a_j s^{j-i}}{\sum_{j=0}^{n} a_j s^j}, \tag{2.1}$$

where $B(s) = \mathcal{L}\{b(t)\}$.

We see that if $b(t)$ is such that $B(s)$ is a rational function, then $F(s)$ will also be a rational function. Thus, the problem is reduced to finding the inverse Laplace transform of $F(s)$. What might be done when $b(t)$ is such that $B(s)$ is not a rational function will be discussed later.

The method employed here to find the inverse Laplace transform of (2.1) is, first, to factor the denominator; next, to expand the rational function into partial fractions; and finally, by applying the linearity property of the inverse Laplace transform, to find the inverse Laplace transform of each of the partial fractions. The algorithms presented in the next three chapters accomplish each of these steps by using elementary concepts of polynomial ring theory, and recursion where possible.

Chapter III

Factoring

The form of the inverse Laplace transform of a rational function depends on the factors in the denominator polynomial. The problem of factoring polynomials is an important area of numerical analysis. The algorithm presented here relies on elementary algebraic considerations. It should be pointed out that any other factoring algorithm could be used with the rest of the programs listed in the appendix by providing suitable interfacing software.

The algorithm presented here is based on the root-finding method of D. E. Muller [2], a brief description of which follows. Given a polynomial, p, start with three initial estimates for a root: x_{-2}, x_{-1}, and x_0. At the nth stage $(0 \leq n \in \mathrm{N})$, fit a quadratic equation to the points $(x_{n-2}, p(x_{n-2})), (x_{n-1}, p(x_{n-1}))$, and $(x_n, p(x_n))$. Then find the root of this quadratic equation nearest x_n. This root becomes x_{n+1}. Repeat this procedure while n is less than a given upper bound, or subsequent estimates fail to improve by a given small amount. To find other roots, deflate p by the appropriate factor and reapply Muller's method if necessary.

Suppose we are given a polynomial with real coefficients and distinct complex roots. Apply Muller's method to obtain a root. Deflate the polynomial by a factor containing the root just found (and its complex conjugate, if necessary) to obtain a polynomial of smaller degree with real coefficients and distinct complex roots. Repeat this process until the degree of the deflated polynomial is of degree 0, at which point all the roots of the original polynomial have been found.

The above process works quite well if all the roots are distinct. If Muller's method is applied to find a multiple root, it converges more more slowly than it does for a single root. That is, it runs through many more iterations before the difference between successive estimates becomes as small. So we turn to abstract algebra for a way to speed things up. By using the method described below, we can ensure that we will always be searching for distinct roots.

Consider a monic polynomial $p \in \mathbf{R}[x]$, with $r \in \mathbf{C}$ a root of p. Then there exists $q \in \mathbf{C}[x]$ such that $p = (x - r)q$. Now consider the first derivative of p, $p' = q + (x - r)q'$. Observe that $p'(r) = 0$ if and only if $q(r) = 0$. Thus r appears more than once as a root of p if and only if $p'(r) = 0$ [3].

Now look at $g_1 = \gcd(p, p') \in \mathbf{R}[x]$. If $\deg(g_1) = 0$ then the roots of p are all distinct. If $\deg(g_1) > 0$ then each root of g_1 is also a root of p and appears more than once as a root of p. Now define $g_n = \gcd(p, p^{(n)})$. If $\deg(g_n) > 0$ then each root of g_n is also a root of p and its multiplicity is at least $n + 1$.

Since the multiplicity of any root is at most $\deg(p)$ there exists a minimal $k \in \mathrm{N}$ such that $\deg(g_k) = 0$. This k can be found by repeated differentiation of p and application of the Euclidean algorithm to obtain each gcd. The first gcd obtained with degree 0 is g_k.

The only roots of g_{k-1} will be all the roots of p with multiplicity k. These roots can be found using Muller's method on g_{k-1}. These roots will be distinct in g_{k-1} but of multiplicity two in g_{k-2}. But the remaining roots of g_{k-2} will be distinct, and can be found by deflating g_{k-2} by all known roots and then applying Muller's method on the deflated polynomial. Similarly, all roots of p of multiplicity n or higher will be roots of g_{n-1}. Say a root has multiplicity $m \geq n$; this root will have multiplicity $m - n + 1$ in g_{n-1}. Thus, we can find all roots of p by working backward from g_k to p using Muller's method and deflation. Most importantly, Muller's method is never used to search for a root of multiplicity greater than 1.

It should be pointed out that repeated polynomial divisions occur in the Euclidean algorithm. A loss of precision is inherent in this process. If one cannot make this sacrifice of accuracy of the results in favor of increased rate of convergence, realize that any other factoring algorithm will work with the rest of the programs in the appendix. The interfacing software would be easy to write.

Chapter IV

Partial Fraction Expansion

One objective in designing a partial fraction expansion algorithm was to minimize the amount of calculation done. Chin and Steiglitz [4] devised an algorithm capable of accomplishing the expansion in $N(N-1)$ multiplications and $\frac{3}{2}N(N-1)$ additions, where N is the degree of the denominator of the given rational function. This algorithm has a disadvantage: it requires use of complex arithmetic. Chin and Steiglitz count complex divisions as equivalent in time to complex multiplications. While this may be true for real divisions and multiplications it certainly is not true for complex ones. Observe that,

$$\frac{a+ib}{c+id} = \frac{(ac+bd)+i(bc-ad)}{c^2+d^2},$$

has 6 real multiplications, 2 real divisions and 3 real additions. Call this 8 multiplications and 3 additions. Furthermore,

$$(a+ib)(c+id) = (ac-bd)+i(bc+ad),$$

requires 4 real multiplications and 2 real additions. Finally, note that

$$(a+ib)+(c+id) = (a+c)+i(b+d),$$

consists of 2 real additions.

Examination of Chin's and Steiglitz's algorithm reveals that the expansion actually involves $\frac{3}{2}N(N-1)$ complex additions, $\frac{1}{2}N(N-1)$ complex multiplications, and $\frac{1}{2}N(N-1)$ complex divisions. Using the above calculations, this results in $6N(N-1)$ real multiplications and $\frac{11}{2}N(N-1)$ additions. So if complex arithmetic can be avoided and the operation count can be held lower than this, the algorithm will be improved.

It is clear that the reason Chin and Steiglitz chose to work with complex numbers is that a polynomial in $\mathbf{R}[x] \subset \mathbf{C}[x]$, splits in $\mathbf{C}$. This makes the partial fraction expansion algorithm simple to describe and analyze. But $\mathbf{R}[x]$ has the property that an irreducible element is either linear or quadratic. If we use this property we can avoid complex arithmetic and thus reduce the number of calculations at the cost of complicating the algorithm a bit.

The key to adapting the algorithm of Chin and Steiglitz is finding a nice way to generalize the following problem. Let $s, r, K \in \mathbf{C}$, for $s \neq r$ and find $A, A^* \in \mathbf{C}$, such that

$$\frac{1}{x-s}\left[\frac{K}{(x-r)^n}\right] = \frac{A^*}{(x-s)(x-r)^{n-1}} + \frac{A}{(x-r)^n}.$$

This problem generalizes to: Let $s, r \in \mathbf{R}[x] \setminus \mathbf{R}$, with $\gcd(s,r) = 1$; and $K \in \mathbf{R}[x]$, with $\deg(k) < \deg(r)$. Find $A, A^* \in \mathbf{R}[x]$ such that $\deg(A) < \deg(r)$, and $\deg(A^*) < \deg(s)$, and

$$\frac{1}{s}\left[\frac{K}{r^n}\right] = \frac{A^*}{sr^{n-1}} + \frac{A}{r^n}. \tag{4.1}$$

First multiply through by r^{n-1} to reduce the problem to:

$$\frac{1}{s}\left[\frac{K}{r}\right] = \frac{A^*}{s} + \frac{A}{r}. \tag{4.2}$$

We know that since $\gcd(s,r) = 1$, there exists $a, b \in \mathbf{R}[x]$ such that $as + br = 1$ and so $K = K[as + br] = Kas + Kbr$. Apply the division algorithm to obtain the following:

$$\begin{aligned} Ka = rq + A \quad &\text{such that} \quad \deg(A) < \deg(r), \\ Kb = sq^* + A^* \quad &\text{such that} \quad \deg(A^*) < \deg(s). \end{aligned} \tag{4.3}$$

Now write

$$\begin{aligned} K = Kas + Kbr &= (rq + A)s + (sq^* + A^*)r \\ &= (q + q^*)rs + As + A^*r. \end{aligned}$$

By hypothesis we have $\deg(K) < \deg(r) < \deg(rs)$ and from (4.3) we get $\deg(As) < \deg(rs)$ and $\deg(A^*r) < \deg(rs)$. So $q + q^* = 0$ and $K = As + A^*r$, which yields (4.2) as required.

Now we must determine how to calculate A and A^* in (4.1) [5], and the necessary number of calculations. First consider the following adaption of the Euclidean algorithm. Let $s, r \in \mathbf{R}[x]$. The division algorithm gives:

$$\begin{aligned}
s &= rq_1 + x_1 && \deg(x_1) < \deg(r) \\
r &= x_1 q_2 + x_2 && \deg(x_2) < \deg(x_1) \\
x_1 &= x_2 q_3 + x_3 && \deg(x_3) < \deg(x_2) \\
&\vdots && \vdots \\
x_{n-2} &= x_{n-1} q_n + x_n && \deg(x_n) < \deg(x_{n-1})
\end{aligned}$$

And also define $x_0 = r$. If, furthermore:

$$\begin{aligned}
a_0 &= 0 && b_0 = 1 \\
a_1 &= 1 && b_1 = -q_1 \\
&\vdots && \vdots \\
a_n &= a_{n-2} - q_n a_{n-1} && b_n = b_{n-2} - q_n b_{n-1}
\end{aligned}$$

then $a_n s + b_n r = x_n$, for all $n \geq 0$.

Proof: $a_0 s + b_0 r = r = x_0$. $a_1 s + b_1 r = s - rq_1 = x_1$. Assume the hypothesis is true for $n-2$ and $n-1$ for some $n \in \mathbf{N}$.

$$\begin{aligned}
x_n &= x_{n-2} - q_n x_{n-1} \\
&= a_{n-2}s + b_{n-2}r - q_n(a_{n-1}s + b_{n-1}r) \\
&= (a_{n-2} - q_n a_{n-1})s + (b_{n-2} - q_n b_{n-1})r \\
&= a_n s + b_n r,
\end{aligned}$$

as claimed.

We can use the above procedure to develop a method for evaluating A and $-A^*$. There are four cases to consider since either s or r can be of degree 1 or 2. Let us begin with the most difficult case:

Case 1: Both s and r are quadratic

We have $s = rq_1 + x$ and $r = x_1q_2 + x_2$. x_2 is a unit so $a_2s + b_2r = -q_2s + (1 + q_1q_2)r = x_2$, and we can get the following.

$$\begin{aligned}\frac{K}{sr} &= \frac{\frac{1}{x_2}K(-q_2s + (1 + q_1q_2)r)}{sr} \\ &= \frac{-\frac{1}{x_2}q_2K}{r} + \frac{\frac{1}{x_2}(1 + q_1q_2)K}{s}.\end{aligned}$$

By the division algorithm, $q_2K = Qr - x_2A$ for some $Q \in \mathbf{R}[x]$. The quotient, Q, does not matter as was shown in the derivation of (4.1). The remainder is the important thing. Now we need to add up all the calculations required to evaluate A.

Table (4.1) Summary of Case 1

To obtain x_1 requires			2 adds,
to obtain q_2 and x_2 requires	2 divs	2 mult	2 adds,
to obtain q_2K requires		4 mults	2 adds,
to divide by r to get x_2A requires		2 mults	2 adds,
and to evaluate A requres	2 divs		1 add.
Resulting in	4 divs	8 mults	9 adds.

Now we have A. We need $-A^*$ as well. Since we know that $K = As + A^*r$, write $K = k_1x + k_0$, $A = a_1x + a_0$, $A^* = a_1^*x + a_0^*$, $s = x^2 + s_1x + s_0$, and $r = x^2 + r_1x + r_0$. We obtain

$$k_1x + k_0 = (a_1x + a_0)(x^2 + s_1x + s_0) + (a_1^*x + a_0^*)(x^2 + r_1x + r_0).$$

Equating coefficients for cubes and constants results in $-a_1^* = a_1$, and $-a_0^* = (a_0s_0 - k_0)/r_0$. So $-A^*$ can be calculated using one multiplication, one division, and one addition. All told, calculation of A and $-A^*$ requires the equivalent of 14 multiplications and 11 additions.

Case 2: s **is quadratic and** r **is linear**

We have $s = rq_1 + x_1$. Since x_1 is a unit, we stop.

$$\frac{K}{sr} = \frac{\frac{1}{x_1}K(s - q_1 r)}{sr}$$
$$= \frac{\frac{K}{x_1}}{r} - \frac{\frac{1}{x_1}q_1 K}{s}$$

$\frac{K}{x_1} = A$, and both K and x_1 are units, so

Table (4.2) Summary of Case 2

to obtain x_1 requires		1 mult	2 adds,
and to obtain A requires	1 div.		
The result is	1 div	1 mult	2 adds.

Now we have, as before,

$$k_1 x + k_0 = A(x^2 + s_1 x + s_0) + (a_1^* x + a_0^*)(x + r_0).$$

Equating coefficients of squares and constants gives $-a_1^* = A$ and $-a_0^* = (As_0 - k_0)/r_0$. So $-A^*$ is computed with one multiplication, one division, and one addition. All together, calculation of A and $-A^*$ requires the equivalent of **4 multiplications and 3 additions.**

Case 3: s **is linear and** r **is quadratic**

We have $s = rq_1 + x_1$, and $r = x_1 q_2 + x_2$. But $q_1 = 0$, thus $x_1 = s$ and notice that x_2 is then a unit, so $-q_2 s + r = x_2$ and then

$$\frac{K}{sr} = \frac{\frac{1}{x_2}K(-q_2 s + r)}{sr}$$
$$= \frac{-\frac{1}{x_2}q_2 K}{r} + \frac{\frac{1}{x_2}K}{s}.$$

This time we will compute $-A^*$, a unit, first. Observe that $K = Qs - (x_2(-A^*))$. Now we do as before and write $k_1 x + k_0 = (a_1 x + a_0)(x + s_0) +$

$A^*(x^2 + r_1 x + r_0)$. Equate coefficients of squares and constants to get $a_1 = -A^*$ and $a_0 = (k_0 - A^* r_0)/s_0$. Now we summarize.

Table (4.3) Summary of Case 3

To obtain x_2 requires		1 mult	2 adds,
to obtain $x_2 A^*$ requires		1 mult	2 adds,
to obtain $-A^*$ requires	1 div		1 add,
and to obtain a_0 requires	1 div	1 mult	1 add.
The result is	2 divs	3 mults	5 adds.

All together calculation of A and $-A^*$ requires the equivalent of **5 multiplications and 5 additions.**

Case 4: r and s are both linear

This case is, of course, the simplest. $s = q_1 r + x_1$, x_1 is a unit and $q_1 = 1$. So

$$\frac{K}{sr} = \frac{\frac{1}{x_1}K(s-r)}{sr}$$
$$= \frac{\frac{K}{x_1}}{r} + \frac{-\frac{K}{x_1}}{s}.$$

Thus $A = -A^* = K/x_1$, which requires **1 multiplication and 1 addition.** Now that each case has been examined, the following table summarizes the preceding information:

Table (4.4) Number of operations required to evaluate (4.2)

deg(s)	deg(r)	multiplications	additions
1	1	1	1
2	1	4	3
1	2	5	5
2	2	14	11

The following is essentially Chin's and Steiglitz's algorithm in $\mathbf{R}[x]$ instead of $\mathbf{C}[x]$. Let $p \in \mathbf{N}$ be given and $d_j \in \mathbf{R}[x]$, for each $1 \leq j \leq p$ and each d_j irreducible over $\mathbf{R}$. Let $m_j \in \mathbf{N}$ denote the multiplicity of each d_j. Also let $Q_0, D \in \mathbf{R}[x]$ be given such that $D = \prod_{j=1}^{p} (d_j)^{m_j}$ and $\deg(Q_0) < \deg(D) = \sum_{j=1}^{p} \deg(d_j) m_j$. Thus we have a proper rational function and wish to find $K_{ij} \in \mathbf{R}[x]$ such that

$$\frac{Q_0}{D} = \frac{Q_0}{\prod_{j=1}^{p} (d_j)^{m_j}} = \sum_{j=1}^{p} \sum_{i=1}^{m_j} \frac{K_{ij}}{(d_j)^i}. \tag{4.4}$$

Define $m_0 = 0$ and $n = \sum_{j=0}^{p} m_j$. Now, for $1 \leq l \leq n$, define

$$u_l = 1 + \min\{x \in \mathbf{N} \cup \{0\} \mid \sum_{j=0}^{x} m_j < l\},$$

$$v_j^l = \begin{cases} m_j, & \text{if } j < u_l; \\ l - \sum_{i=0}^{u_l - 1} m_i, & \text{if } j = u_l, \end{cases}$$

for $1 \leq j \leq u_l$. Also, for each l, define $f_l = d_{u_l}$, and $R_l, Q_l \in \mathbf{R}[x]$ such that $Q_{l-1} = Q_l f_l + R_l$. Again define $A_{lj}(K), A_{lj}^*(K) : \mathbf{R}[x] \mapsto \mathbf{R}[x]$ by

$$\frac{K}{f_l d_j} = \frac{A_{lj}^*(K)}{f_l} + \frac{A_{lj}(K)}{d_j}.$$

Now the partial fraction expansion can be obtained as below in n steps, the lth step being:

$$\begin{aligned} \frac{Q_0}{\prod_{j=1}^{p} d_j^{m_j}} &= \frac{1}{\prod_{j=1}^{n} f_j}[Q_0] \\ &= \frac{1}{\prod_{n \geq k > l} f_k}[Q_l + \sum_{j=1}^{u_l} \sum_{i=1}^{v_j^l} \frac{K_{ij}^l}{(d_j)^i}], \end{aligned} \tag{4.5}$$

where it is understood that $\prod_{n\geq k>n} f_k = 1$ and

$$K_{ij}^l = \begin{cases} 0, & \text{if } i > v_j^l \text{ or } j > u_l \text{ or } l = 0; \\ A_{lj}(K_{ij}^{l-1}), & \text{if } j < u_l \text{ and } i = v_j^l; \\ A_{lj}(K_{ij}^{l-1} + A_{lj}^*(K_{(i+1)j}^{l-1})), & \text{if } j < u_l \text{ and } i < v_j^l; \\ K_{(i-1)j}^{l-1}, & \text{if } j = u_l \text{ and } i > 1; \\ R_l + \sum_{0\leq k<1} A_{lk}^*(K_{1k}^{l-1} + A_{lk}^*(K_{2k}^{l-1})), & \text{if } j = u_l \text{ and } i = 1. \end{cases} \tag{4.6}$$

Now we must count operations to compare this method with Chin's and Steiglitz's method. It turns out that the number of operations depends on the number of quadratic factors in the denominator of the rational function. Let $N = \deg(D)$ and then denote the number of quadratic factors in D as q. Thus $N = n + q$.

First consider the number of calculations necessary to compute $\{R_l \mid 1 \leq l \leq n\}$. The lth stage involves dividing Q_{l-1} by f_l. Two facts are necessary: To divide an M-degree polynomial by a monic linear factor requires M multiplications and M additions and to divide the same polynomial by a monic quadratic factor requires $2(M-1)$ multiplications and additions. Note that the largest that $\deg(Q_0)$ can be is $N-1$. We shall prove that in this worst case it requires no more than $\frac{1}{2}N(N-1)-q$ multiplications and additions to compute $\{R_l \mid 1 \leq l \leq n\}$.

We shall use induction on N. For $N = 1$ we must have $n = 1$, and $q = 0$, and of course, $\deg(Q_0) = 0$ thus $\frac{1}{2}N(N-1) - q = 0$, which reflects the fact that there is really nothing to do in this case. We will also need to examine the case where $N = 2$, with $n = 1$ and $q = 1$. We still get $\frac{1}{2}N(N-1) - q = 0$, which again indicates that there is really no partial fraction expansion to carry out. Now assume the result for a given N.

First assume that we add a linear factor to D and increase $\deg(Q_0)$ to $(N+1) - 1 = N$. So divide Q_0 by this new linear factor to get $\deg(Q_1) = N - 1$, which will require N multiplications and additions. Now apply the induction hypothesis to Q_1. It will require $\frac{1}{2}N(N-1) - q$ multiplications and additions to obtain $\{R_l \mid$

$2 \leq l \leq n+1\}$. Adding up, we get

$$N + \frac{1}{2}N(N-1) - q = \frac{1}{2}(N+1)N - q.$$

Now assume that we add a quadratic factor to D and increase $\deg(Q_0)$ to $(N+2) - 1 = N+1$. Dividing Q_0 by the new quadratic factor requires $2N$ multiplications and additions. We are left with $\deg(Q_1) = N-1$. By induction, to compute $\{R_l \mid 2 \leq l \leq n\}$ requires $\frac{1}{2}N(N-1) - q$ multiplications and additions. Summing, we get

$$2N + \frac{1}{2}N(N-1) - q = \frac{1}{2}(N+2)(N+1) - (q+1),$$

as required.

We must also consider the necessary number of calculations required to compute $\{K_{ij}^l \mid 1 \leq j \leq u_l, \text{and} 1 \leq i \leq v_j^l\}$ for some $1 \leq l \leq n$. It turns out that the number of operations needed to compute this set depends on $\deg(d_{u_l})$, on m_{u_l}, and on the number of quadratic factors preceding d_{u_l}

Notice that computation of $K_{iu_l}^l$ requires no calculation for $i > 1$. This means that the largest operation count for the partial fraction expansion algorithm occurs when all the factors of D are distinct, that is, when $m_j = 1$ for all $1 \leq j \leq p = n$. Consequently, $u_l = l$, so $f_l = d_l$ and $v_j^l = m_j = 1$ for all admissable l and j. Now write (4.5) as

$$\frac{Q_0}{\prod_{k=1}^{n} f_k} = \frac{1}{\prod_{n \geq k > l} f_k}[Q_l + \sum_{j=1}^{l} \frac{K_{1j}^l}{f_j}].$$

And we can also write (4.6) as

$$K_1^l j = \begin{cases} 0, & \text{if } l = 0 \\ A_{lj}(K_{1j}^{l-1}), & \text{if } 1 < j < l \\ R_l + \sum_{0 \leq k < l} A_{lk}^*(K_{1k}^{l-1}), & \text{if } j = l. \end{cases}$$

Calculation of $\{K_{1j}^l \mid 1 \leq j \leq l\}$ for a given l requires that A_{lj} and $-A_{lj}^*$ be determined $l-1$ times along with $(l-1)\deg(f_l)$ additions. Consider the number of calculations in computation of A_{lj} and A^*_{lj}. Table (4.4) gives us this information if we let $s = f_l$ and $r = f_j$. Notice that, given f_l, it will take the most operations if f_j is quadratic. Hence, the number of calculations in computing $\{K^l_{1j} \mid 1 \leq j \leq l\}$ will be largest if $\deg(f_k) = 2$ for all $k \leq l$.

In order to find an upper bound on the number of calculations in this algorithm, assume all factors of D are distinct and ordered such that all quadratic factors appear first. Then the entire algorithm would require

$$\begin{aligned}\frac{1}{2}N(N-1) - q + \sum_{k=1}^{q} 14(k-1) + \sum_{q+1}^{n}(5q + (k-(q+1)) \\ = N(N-1) - q^2 + 3Nq - 7q\end{aligned} \tag{4.7}$$

multiplications. This formula also works for $q = 0$ and $q = N/2$. The result in each case is $N(N-1)$ and $\frac{9}{4}N(N-2)$, respectively which is easily verified. Also the algorithm will require

$$\begin{aligned}\frac{1}{2}N(N-1) - q + \sum_{k=1}^{q}(11+2)(k-1) \\ + \sum_{q+1}^{n}((5+2)q + (1+1)(k-(q+1)) \\ = \frac{3}{2}N(N-1) - \frac{7}{2}q^2 + 3Nq - \frac{11}{2}q\end{aligned} \tag{4.8}$$

additions. Again, for the special cases $q = 0$ and $q = N/2$, the formula yields $\frac{3}{2}N(N-1)$ and $\frac{17}{4}N(N-2)$ respectively. In fact it is quite easy to show that for all $N \geq 2$ and $N/2 \leq q \leq 0$ we get the following:

$$6N(N-1) > \frac{9}{4}N(N-2) \geq N(N-1) - q^2 + 3Nq - 7q.$$

This shows that the greatest number of multiplications occurs when $q = n$, and is still less than the number required by Chin's and Steiglitz's algorithm. Also observe

$$\frac{11}{2}N(N-1) > \frac{17}{4}N(N-2) \geq \frac{3}{2}N(N-1) - \frac{7}{2}q^2 + 3Nq - \frac{11}{2}q.$$

Which tells the same story for additions.

These results show that this adaption of Chin's and Steiglitz's algorithm saves calculations. To be fair, however, one must realize that the output of this adaptation is not the same as that of Chin and Steiglitz. Which algorithm is better will depend on the application. Partial fractions expansions can be useful in a wide scope of problems involving integrals of rational functions [6].

Chin's and Steiglitz's output differs from the one presented here in that $\mathbf{C}[x]$ is the ambient polynomial ring and each denominator in the partial fractions expansion is thus linear. With the adapted algorithm, $\mathbf{R}[x]$ is used and the denominators can be linear or quadratic. It happens that in computing inverse Laplace transforms, either form is acceptable and it is better to have fewer operations; however, this may not always be so for other applications of partial fraction expansion.

Chapter V

Inverse Laplace Transform

The most elementary approach to finding the inverse Laplace transform of a given function is to use a table of transform pairs. Indeed, large tables of transform pairs have been prepared. In particular, a popular table [7] lists the following transform pair:

$$\mathcal{L}^{-1}\left\{\frac{1}{(s^2+a^2)^k}\right\} = \frac{\sqrt{\pi}}{\Gamma(k)}\left(\frac{t}{2a}\right)^{k-1/2} J_{k-1/2}(at), \qquad k \in \mathbb{N}. \tag{5.1}$$

The functions J_p are known as the Bessel functions of half-integral order. They have the following recursive definition [8]:

$$\begin{aligned} J_{-1/2}(t) &= \sqrt{\frac{2}{\pi t}}\cos t, \\ J_{1/2}(t) &= \sqrt{\frac{2}{\pi t}}\sin t, \\ J_{p+1}(t) &= \frac{2p}{t}J_p(t) - J_{p-1}(t). \end{aligned}$$

Let us simplify things somewhat by defining:

$$H_k(at) = \sqrt{\pi}\left(\frac{t}{2a}\right)^{k-1/2} J_{k-1/2}(at).$$

We thus obtain the recursive relationship:

$$\begin{aligned} H_0(at) &= \frac{2}{t}\cos(at), \\ H_1(at) &= \frac{1}{a}\sin(at), \\ H_{k+1}(at) &= \frac{2k-1}{2a^2}H_k(at) - \left(\frac{t}{2a}\right)^2 H_{k-1}. \end{aligned}$$

Hence the Laplace transform pair (5.1) becomes:

$$\mathcal{L}^{-1}\left\{\frac{1}{(s^2+a^2)^k}\right\} = \frac{1}{\Gamma(k)}H_k(at).$$

If we use a well-known property of the Laplace transform, we can derive another important Laplace transform pair. Use

$$\mathcal{L}^{-1}\left\{\frac{d}{ds}F(s)\right\} = -tf(t)$$

to get

$$\mathcal{L}^{-1}\left\{\frac{d}{ds}\left(\frac{1}{(s^2+a^2)^k}\right)\right\} = -\frac{t}{\Gamma(k)}H_k(at)$$

$$\mathcal{L}^{-1}\left\{\frac{-2ks}{(s^2+a^2)^{k+1}}\right\} = -\frac{t}{\Gamma(k)}H_k(at)$$

$$\mathcal{L}^{-1}\left\{\frac{s}{(s^2+a^2)^{k+1}}\right\} = \frac{t}{2\Gamma(k+1)}H_k(at).$$

Which is equivalent to

$$\mathcal{L}^{-1}\left\{\frac{s}{(s^2+a^2)^{k}}\right\} = \frac{t}{2\Gamma(k)}H_{k-1}(at).$$

This would be true for $k \geq 2$ but also holds for $k = 1$. Again make use of a Laplace transform property, namely $\mathcal{L}^{-1}\{F(s+\tau)\} = e^{-\tau t}f(t)$ to get one of the Laplace transform pairs useful in this problem:

$$\mathcal{L}^{-1}\left\{\frac{A(s+\tau)+B}{\left((s+\tau)^2+a^2\right)^k}\right\} = \frac{e^{-\tau t}}{\Gamma(k)}\left[BH_k(at) + \frac{At}{2}H_{k-1}(at)\right]. \tag{5.2}$$

We also make use of another often-tabulated Laplace transform pair [7]:

$$\mathcal{L}^{-1}\left\{\frac{A}{(s+\tau)^k}\right\} = \frac{Ae^{-\tau t}t^{k-1}}{\Gamma(k)}. \tag{5.3}$$

One of these two Laplace transform pairs will apply to each term in the partial fraction expansion of a rational function. From (4.3) and we have

$$\mathcal{L}^{-1}\left\{\frac{Q_0}{\prod_{j=1}^{p}(d_j)^{m_j}}\right\} = \mathcal{L}^{-1}\left\{\sum_{j=1}^{p}\sum_{i=1}^{m_j}\frac{K_{ij}}{(d_j)^i}\right\}$$

$$= \sum_{j=1}^{p}\sum_{i=1}^{m_j}\mathcal{L}^{-1}\left\{\frac{K_{ij}}{(d_j)^i}\right\}. \tag{5.4}$$

Now if $\deg(K_{ij}) = 1$ use (5.3), and if $\deg(K_{ij}) = 2$ use (5.2).

Using (5.2), and induction it can be shown that:

$$\mathcal{L}^{-1}\left\{\frac{A(s+\tau)+B}{\left((s+\tau)^2+a^2\right)^k}\right\} = e^{-\tau t}\sum_{j=0}^{k-1} t^j\left(\alpha_j\cos(at)+\beta_j\sin(at)\right).$$

And also note that (5.3) can be written

$$\mathcal{L}^{-1}\left\{\frac{A}{(s+\tau)^k}\right\} = e^{-\tau t}t^{k-1}\left(A\cos(0)\right).$$

Thus we can express (5.4) in the form:

$$\sum_{j=1}^{p}\sum_{i=1}^{m_j}\mathcal{L}^{-1}\left\{\frac{K_{ij}}{(d_j)^i}\right\} = \sum_{j=1}^{p} e^{-\tau_j t}\sum_{i=1}^{m_j} t^i\left(\alpha_{ji}\cos(a_j t)+\beta_{ji}\sin(a_j t)\right). \tag{5.5}$$

This is the way the inverse Laplace transform is computed by the program in the appendix. The output is simply an array of coefficients for an expression of the form (5.5).

Chapter VI

Applications

One of the most probable applications of this algorithm will be to evaluate transient responses of control systems with a known transfer function. Figure (6.1) shows a control diagram for an automatic flight control system for a supersonic aircraft [9]. The transfer function corresponding to figure (6.1) will depend on the values of K_1 and K_2. It is simple to compute this transfer function using the coefficient values shown in figure (6.1).

The above transfer function was inverse transformed using various values for K_1 and K_2. The transient response functions so obtained are graphed in figures (6.2) and (6.3). Notice how the response improves until the onset of instability.

It was mentioned before that sometimes we want the inverse transform of something other than a rational function. One common example arises when a control system contains time delays. We have the following transform pair.

$$\mathcal{L}\left\{u(t-T)\right\} = \frac{e^{-sT}}{s}$$

In order to apply the algorithm, we must approximate e^{sT} by a rational function. This can be done using the Padé approximant [9]. We have

$$e^{-sT} \simeq P_n(sT) = \frac{\sum_{j=0}^{n} (-1)^j b_j (sT)^j}{\sum_{j=0}^{n} b_j (sT)^j},$$

where

$$b_j = \frac{\binom{n}{j}}{\binom{2n}{j} j!}.$$

Using the Padé approximant, the algorithm was used to invert $\frac{e^{-s}}{s}$. Padé approximants of order 2,3,4, and 6 were used. These approximants are shown graphically in figure (6.4).

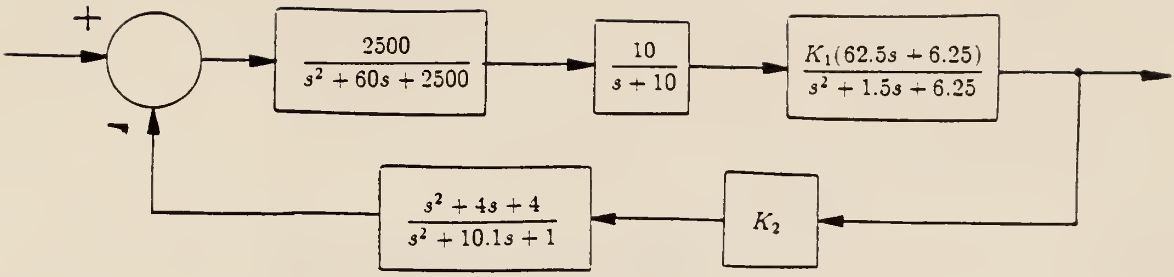

Figure (6.1) Control system diagram of SST aircraft. Adapted (9).

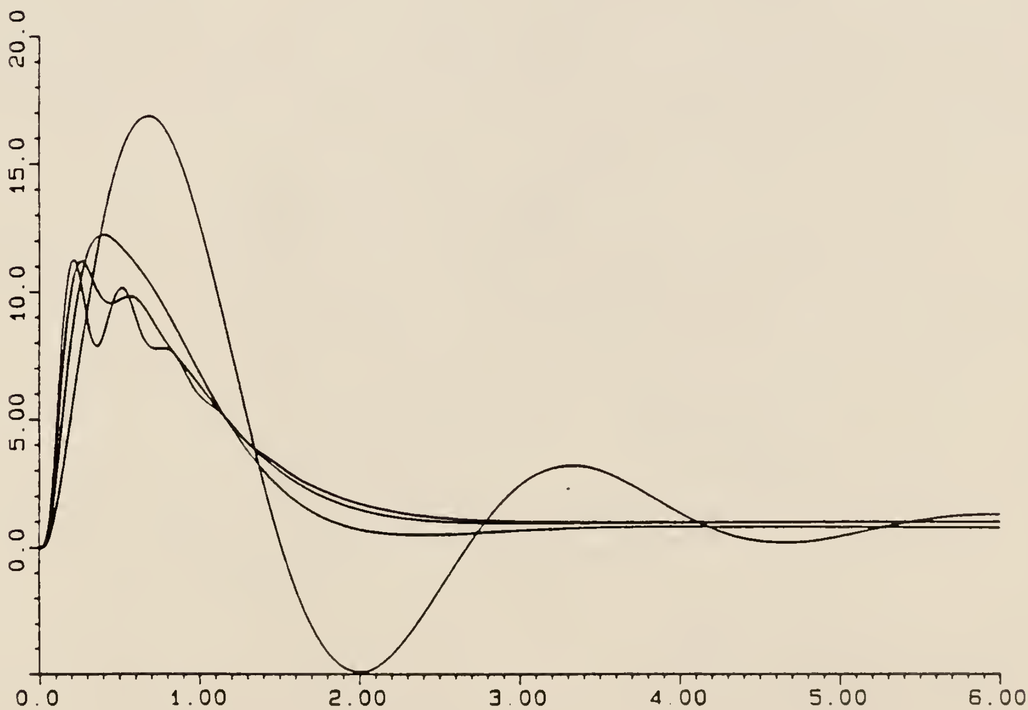

Figure (6.2) Step response of above control system for $K_1 K_2 = 0, 0.2, 0.4$, and 0.6.

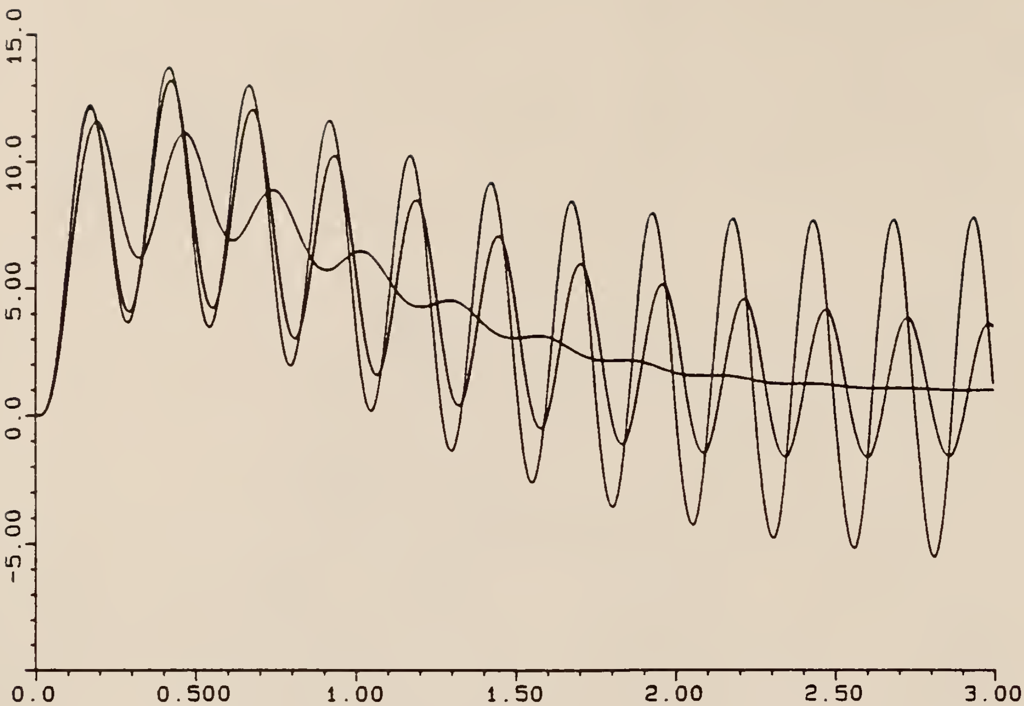

Figure (6.3) Step response of above control system for $K_1K_2 = 0.8, 1.08$, and 1.10.

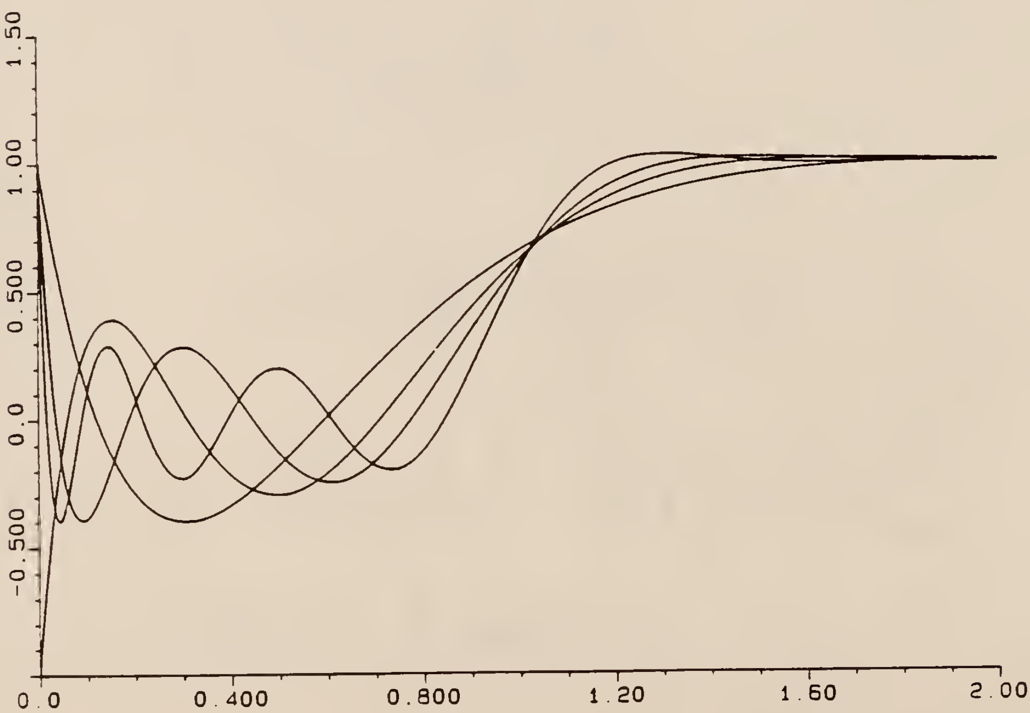

Figure (6.4) Padé approximants to unit time delay.

Chapter VII

Conclusion

This paper has presented an algorithm for computing inverse Laplace transforms of rational functions as might arise in practical electrical engineering problems. Programs written to demonstrate the algorithm follow in the appendix.

Numerical analysis aspects of the problem were not dealt with, but, except for root-finding, the problem was shown to be an algebraic one. Results from elementary abstract algebra were used to derive the methods described. Special effort was made to reduce the number of calculations in the partial fraction expansion.

Some applications were presented to show practical results. These applications also made it clear that assuming that the Laplace domain function, $F(s)$, in (2.1) is a ratio of polynomials is not always valid. Future work on this problem should concern itself with this assumption.

Appendix I

List of References

[1] Ross, S. L. *Introduction to Ordinary Differential Equations*, pp. 427-458, New York:John Wiley & Sons, 1980.

[2] Muller, D. E. "A Method for Solving Algebraic Equations Using an Automatic Computer," *Mathematical Tables and Other Aids to Computation*, vol. 10, pp. 208-215, 1956.

[3] Herstein, I. N. *Topics in Algebra*, p. 233, New York:John Wiley & Sons, 1975.

[4] Chin, F. Y., and Steiglitz, K. "An O(N^2) Algorithm for Partial Fraction Expansion" *IEEE Transactions on Circuits and Systems*, vol. 24, pp. 42-45, 1977.

[5] Hamming, R. W. *Calculus and the Computer Revolution*, pp. 49-51, Boston:Houghton Mifflin, 1968.

[6] Birkhoff, and Mac Lane *A Survey of Modern Algebra*, pp. 17-19 and 103-105, New York:MacMillan, 1941.

[7] Beyer, W. H. (editor) *CRC Standard Mathematical Tables*, pp. 469, Boca Raton, FL:CRC Press, 1984.

[8] Simmons,G. F. *Differential Equations with Applications and Historical Footnotes*, pp. 242-243, New York:McGraw-Hill, 1972.

[9] Dorf, R. C. *Modern Control Systems*, pp. 186-187, Reading, MA:Addison-Wesley, 1974.

[10] Hausner, A. *Analog and Analog/Hybrid Computer Programming*, pp. 275-278 and 282-283, Englewood Cliffs, NJ:Prentice Hall, 1971.

[11] Press, W. H., *et al.* *Numerical Recipes*, pp. 137-140 and 259-262, Cambridge, U.K.:Cambridge University Press, 1986.

Appendix II

Glossary of Terms

C denotes the field of complex numbers.

C[x] denotes the ring of polynomials with coefficients in **C**.

deg: If $p(x) = a_0 + a_1x + \cdots + a_nx^n \neq 0$ and $a_n \neq 0$ then $\deg(p(x)) = n$.

Division algorithm: Given two polynomials $p(x), q(x) \in F[x]$, where $F[x]$ is the ring of polynomials with coefficients in the field F and $q(x) \neq 0$, there exist two polynomials $t(x), r(x) \in F[x]$ such that $f(x) = t(x)q(x) + r(x)$ where $r(x) = 0$ or $deg(r(x)) < deg(q(x))$. The process by which $t(x)$ and $r(x)$ are found is known as the division algorithm and is simply the "long-division" process everyone knows to divide one polynomial by another.

Euclidean algorithm: Given two polynomials $p(x), q(x) \in F[x]$, where $F[x]$ is the ring of polynomials with coefficients in the field F and $p(x)$ and $q(x)$ are not both 0, then $\gcd(p(x), q(x)) \in F[x]$ exists and there exist polynomials $m(x), n(x) \in F[x]$ such that $\gcd(p(x), q(x)) = m(x)p(x) + n(x)q(x)$. The process used to determine these special polynomials is called the Euclidean algorithm and is shown explicitly in Chapter IV.

Γ: A sufficient definition of the function Γ for $n \in \mathbf{N} \cup \{0\}$ is

$$\Gamma(n) = \begin{cases} 0, & \text{if } n = 0; \\ (n-1)!, & \text{if } n > 0. \end{cases}$$

gcd: Let $a, b \in F[x]$. If $c \in F[x]$ satisfies:

1. c is monic.

2. c divides a and b.

3. Any other divisor of a and b divides c.

then c is called the greatest common divisor of a and b and is denoted $\gcd(a, b)$.

irreducible: Let $p \in F[x]$ be such that $p = ab$ for some $a, b \in F[x]$ if and only if $\deg(a) = 0$ or $\deg(b) = 0$. Such a p is said to be irreducible over F. Note that irreducibility depends on the field, F. $x^2 + 1$ is irreducible over $\mathbf{R}[x]$ but not over $\mathbf{C}[x]$.

min is a function that operates on a well ordered set and whose value is the minimum element of that set. The well ordering property of $\mathbf{N}$ asserts that if $A \subset \mathbf{N}$ then $\min A$ exists.

$\mathbf{N}$ denotes the ring of natural numbers.

$\mathbf{R}$ denotes the field of real numbers.

$\mathbf{R}$[x] denotes the ring of polynomials with coefficients in $\mathbf{R}$. $u(t)$: Let $t \in \mathbf{R}$.

$$u(t) = \begin{cases} 0, & \text{if } t < 0; \\ 1, & \text{if } t \geq 0. \end{cases}$$

unit: $u \in F[x]$ is called a unit iff deg(u)=0.

Appendix III

FORTRAN Programs

The following programs are intended to merely demonstrate the algorithms described in the previous chapters. Specifically, they were used to evaluate the example applications mentioned in Chapter VI. No guarantee of their usefulness to any other application is implied.

These programs could certainly be made more user-friendly. There are no error handling routines, the user interaction is minimal, and file management is cumbersome. Such things are left to a better programmer. Nevertheless, the programs serve their purpose of demonstrating the algorithms.

The programs fall into four slightly overlapping categories: those associated with Chapters III through V and those for creating data files and making plots. The following is a brief categorical index of the programs. Subroutine dependence is indicated by indentions.

Factoring Program

```
ROOT_FIND
    POLY_READ
    FACTORER
        DERIV
        EUCLID
            POLDIV
        FIND_EM
            MULLER
                DEFVAL
                COMPOSE
    SPEC_WRITE
```

Partial Fraction Expansion Program

```
PART_FRAC
    SPEC_READ
    EXPAND
        TRANSFER
        POLDIV
        ALIKE
            TRANSFER
        DIFFERENT
            TRANSFER
            EUCLIDEAN
                TRANSFER
                POLDIV
                POLMULT
            POLADD
    PART_WRITE
```

Inverse Laplace Transform Program

```
INVERT
    INV2
        INIT
        BESSEL
        GAMMA
```

Plotting Program

```
PLOT
    READ
    LOTS_O_PLOTS
    PLOT_O_MATIC
```

Data Entry Routines

```
INPUT_RAT

SPEC_INPUT
    SPEC_WRITE
```

The programs that follow are listed in alphabetical order according to their VAX FORTRAN filenames. Each program is preceded by a header that explains the purpose of the program, and describes the variables passed to and from the program. I have tried to make each header as complete as required to be all the documentation necessary to comprehend the program it describes.

```
*****************************************************************
*       Department of Electrical and Computer Engineering       *
*                   Kansas State University                     *
*                                                               *
*       VAX FORTRAN source filename:    ALIKE.FOR               *
*****************************************************************
*
*       ROUTINE:            SUBROUTINE
*                           ALIKE(I,J,X,DEGX,REM,DEGR)
*
*
*       DESCRIPTION:        Refer to equation (4.6) in the main
*                           thesis.  This program computes K^l_ij
*                           when j=u_l, hence the name ALIKE.
*
*
*       DOCUMENTATION
*       FILES:              None.
*
*
*       ARGUMENTS:          The following arguments are passed to
*                           the subroutine:
*
*           I               (input) integer
*                           corresponds to j in (4.6)
*
*           J               (input) integer
*                           corresponds to i in (4.6)
*
*           X               (input) real
*                           is a three dimensional array, X(I,J,K)
*                           represents to Ith coefficient of the
*                           numerator of the (J,K)th term in the
*                           partial fraction expansion.  Namely,
*                           that term with the Jth factor of DEN
*                           to the Kth power as denominator.
*
*           DEGX            (input) integer
*                           is an array.  DEGX(I,J) represents the
*                           degree of the numerator of the (I,J)th
*                           term in the partial fraction expansion.
*                           See the description of X.
*
*           REM             (input) real
*                           corresponds to R_l in (4.6).
*
*           DEGR            (input) integer
*                           is the degree of R_l in (4.6).
*
*       RETURN:             Not used.
```

```
*
*
*       ROUTINES
*       CALLED:          PUTX
*
*
*       AUTHOR:          James F. Stafford
*
*
*       DATE CREATED:    8Jun87   Version 1.0
*
*
*       REVISIONS:       None.
*
*
**********************************************************************
        SUBROUTINE       ALIKE(I,J,X,DEGX,REM,DEGR)

        IMPLICIT         NONE

        INTEGER          DEGR,I,J,K,L,DEGX(10,*)

        REAL*8           X(0:1,10,*),REM(0:10)

        DO K=J,2,-1

           DEGX(I,K)=DEGX(I,K-1)

           DO L=0,DEGX(I,K)

              X(L,I,K)=X(L,I,K-1)

           ENDDO

        ENDDO

        CALL PUTX(I,1,REM,DEGR,X,DEGX)

        RETURN

        END
```

```
*******************************************************************
*      Department of Electrical and Computer Engineering          *
*                  Kansas State University                        *
*                                                                 *
*      VAX FORTRAN source filename:    BESSEL.FOR                 *
*******************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        BESSEL(F,A,N)
*
*      DESCRIPTION:      This program computes the recursively
*                        defined function H_k described in
*                        chapter V.
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        The following arguments are passed to
*                        the subroutine:
*
*         F              (input) real
*                        is an array containing the coefficients
*                        of the functions H_k.  For a given j,
*                        F(I,0,K) represents the coefficient of
*                        the cosine term of H_(j-I), with t to
*                        the power K.  F(I,1,K) represents the
*                        coefficient of the sine term of H_(j-I),
*                        with t to the power K.
*
*         A              (input) real
*                        represents a in the recursive definition
*                        of H_k in chapter V.
*
*         N              (input) integer
*                        represents k in the recursive definition
*                        of H_k in chapter V.
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           None.
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     9Jun87   Version 1.0
*
```

```
*
*       REVISIONS:      None.
*
*
*************************************************************************
        SUBROUTINE      BESSEL(F,A,N)

        IMPLICIT        NONE

        INTEGER         N,J,K

        REAL*8          A,F(-2:0,0:1,-1:9)

        DO J=-1,N-2

            DO K=-2,-1

                F(K,0,J)=F(K+1,0,J)
                F(K,1,J)=F(K+1,1,J)

            ENDDO

        ENDDO

        DO J=0,N-2

            F(0,0,J)=F(-1,0,J)*(2*N-3)/A
            F(0,1,J)=F(-1,1,J)*(2*N-3)/A

        ENDDO

        DO J=-1,N-3

            F(0,0,J+2)=F(0,0,J+2)-F(-2,0,J)
            F(0,1,J+2)=F(0,1,J+2)-F(-2,1,J)

        ENDDO

        RETURN

        END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                     *
*                                                              *
*      VAX FORTRAN source filename:COMPOSE.FOR                 *
*****************************************************************
*
*      ROUTINE:COMPOSE(X,FACTOR,NUM,DEGF,MULT)
*
*
*      DESCRIPTION:      This program accepts a complex-valued
*                        root,X as input, decides whether X is
*                        purely real or not, and updates the
*                        factor array, FACTOR, accordingly.
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:
*
*          X             (input) complex
*                        Is a complex-valued root of a polynomial.
*
*          FACTOR        (input/output) real
*                        Is an array containing each factor of the
*                        above polynomial.
*
*          NUM           (input/output) integer
*                        Is the number of factors in FACTOR.  NUM
*                        is already incremented before calling
*                        COMPOSE.
*
*          DEGF          (input/output) integer
*                        Is an array specifying the degree of each
*                        corresponding factor in FACTOR.
*
*          MULT          (input/output) integer
*                        Is an array specifying the multiplicity of
*                        each factor in FACTOR.
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           None.
*
*
*      AUTHOR:           James F. Stafford
*
```

```
*
*       DATE CREATED:    30Jun88 Version 1.0
*
*
*       REVISIONS:       None.
*
*
*******************************************************************
        SUBROUTINE       COMPOSE(X,FACTOR,NUM,DEGF,MULT)

        IMPLICIT         NONE

        INTEGER          NUM,DEGF(*),MULT(*)

        REAL*8           FACTOR(10,0:2),SMALL

        COMPLEX*16       X

        LOGICAL          REAL

        PARAMETER        (SMALL=10E-4)

        REAL=.FALSE.

        IF (DREAL(X).NE.0.) THEN

            IF (DABS(DIMAG(X)/DREAL(X)).LT.SMALL) THEN

                REAL=.TRUE.
                FACTOR(NUM,1)=1
                FACTOR(NUM,0)=-DREAL(X)
                DEGF(NUM)=1

            ENDIF

        ELSE IF (CDABS(X).LT.SMALL) THEN

            REAL=.TRUE.
            FACTOR(NUM,1)=1.
            FACTOR(NUM,0)=0.
            DEGF(NUM)=1

        ENDIF

        IF (REAL.EQ..FALSE.) THEN

            FACTOR(NUM,2)=1
            FACTOR(NUM,1)=-2.*DREAL(X)
            FACTOR(NUM,0)=DIMAG(X)*DIMAG(X)+DREAL(X)*DREAL(X)
            DEGF(NUM)=2
```

```
ENDIF
MULT(NUM)=1
RETURN
END
```

```
**********************************************************************
*       Department of Electrical and Computer Engineering           *
*                    Kansas State University                        *
*                                                                   *
*       VAX FORTRAN source filename: DEFVAL.FOR                     *
**********************************************************************
*
*       ROUTINE:            COMPLEX*16 FUNCTION
*                           DEFVAL(POLY,DEG,FACTOR,NUM,DEGF,MULT,X)
*
*
*       DESCRIPTION:        This program evaluates a polynomial at
*                           a given complex argument, X, all known
*                           factors are divided out.
*
*
*       DOCUMENTATION
*       FILES:              None.
*
*
*       ARGUMENTS:          The following arguments are passed to
*                           the function:
*
*           POLY            (input) real
*                           is an array containing coefficients of
*                           the polynomial to be evaluated.
*
*           DEG             (input) integer
*                           is the degree of POLY.
*
*           FACTOR          (input) real
*                           is an array containg the coefficients
*                           of all known factors of POLY.
*
*           NUM             (input) integer
*                           is the number of factors in FACTOR.
*
*           DEGF            (input) integer
*                           is an array specifying the degree of
*                           each corresponding factor in FACTOR.
*
*           MULT            (input) integer
*                           is an array specifying the multiplicity
*                           of each corresponding factor in FACTOR.
*
*           X               (input) complex
*                           is the argument at which the polynomial
*                           is to be evaluated.
*
*
```

```
*       RETURN:          Not used.
*
*
*       ROUTINES
*       CALLED:          None.
*
*
*       AUTHOR:          James F. Stafford
*
*
*       DATE CREATED:    30Jun88 Version 1.0
*
*
*       REVISIONS:       None.
*
*
***************************************************************
      COMPLEX*16 FUNCTION     DEFVAL(POLY,DEG,FACTOR,NUM,DEGF,MULT,X)

      IMPLICIT         NONE

      INTEGER          DEG,I,NUM,DEGF(*),MULT(*)

      REAL*8           POLY(0:*),FACTOR(10,0:2)

      COMPLEX*16       X,EVAL

      DEFVAL=POLY(DEG)

      DO I=DEG-1,0,-1

          DEFVAL=DEFVAL*X+POLY(I)

      ENDDO

      DO I=1,NUM

          IF (DEFVAL.NE.0) THEN

              DEFVAL=DEFVAL/EVAL(FACTOR,I,DEGF(I),MULT(I),X)

          ENDIF

      ENDDO

      RETURN

      END

      COMPLEX*16 FUNCTION     EVAL(FACTOR,I,DEGF,MULT,X)
```

```
IMPLICIT        NONE

INTEGER         J,I,DEGF,MULT

REAL*8          FACTOR(10,0:2)

COMPLEX*16      X,VALUE

EVAL=1
VALUE=FACTOR(I,DEGF)

DO J=DEGF-1,0,-1

   VALUE=VALUE*X+FACTOR(I,J)

ENDDO

DO J=1,MULT

   EVAL=EVAL*VALUE

ENDDO

RETURN

END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering        *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:DERIV.FOR                    *
*****************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        DERIV(POLY,DEG)
*
*
*      DESCRIPTION:      This program computes the derivative of
*                        a given polynomial.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        The following arguments are passed to
*                        the routine.
*
*          POLY          (input/output) real
*                        is an array containing the coefficients
*                        of the polynomial to be differentiated.
*                        On return, this array contains the
*                        coefficients of the derivative.
*
*          DEG           (input/output) integer
*                        is the degree of the polynomial on
*                        input and the degree of the
*                        derivative on output.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           None.
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     30Jun88 Version 1.0
*
*
*      REVISIONS:        None.
*
*
```

```
**********************************************************************
      SUBROUTINE      DERIV(POLY,DEG)

      IMPLICIT        NONE

      INTEGER         I,J,DEG

      REAL*8          POLY(0:*)

      DO J=0,DEG-1

          POLY(J)=(J+1)*POLY(J+1)

      ENDDO

      POLY(DEG)=0.
      DEG=DEG-1

      RETURN

      END
```

```
*****************************************************************
*     Department of Electrical and Computer Engineering        *
*                   Kansas State University                    *
*                                                              *
*     VAX FORTRAN source filename:     DIFFERENT.FOR           *
*****************************************************************
*
*     ROUTINE:          SUBROUTINE
*                       DIFFERENT(I,B,DEGB,DEN,DEGD,MULTS,X,DEGX)
*
*
*     DESCRIPTION:      Refer to equation (4.6) in the main
*                       thesis.  This program computes K^l_ij
*                       when j<u_l, hence the name DIFFERENT.
*
*
*     DOCUMENTATION
*     FILES:            None.
*
*
*     ARGUMENTS:        The following arguments are passed to the
*                       subroutine:
*
*         I             (input) integer
*                       corresponds to j in (4.6)
*
*         B             (input) real
*                       is an array containing the coefficients of
*                       the polynomial f_l in the notation of
*                       chapter IV.
*
*         DEGB          (input) integer
*                       is the degree of B
*
*         DEN           (input) real
*                       is a two-dimensional array.  DEN(I,J)
*                       represents the coefficient of the Ith
*                       power of x in the Jth factor of the
*                       denominator polynomial.
*
*         DEGD          (input) integer
*                       is an array.  DEGD(I) represents the
*                       degree of the Ith factor in the
*                       denominator polynomial.
*
*         MULTS         (input) integer
*                       is an array.  MULTS(I) represents the
*                       multiplicity of the Ith factor in the
*                       denominator polynomial.
*
```

```
*           X               (input) real
*                           is a three-dimensional array.  X(I,J,K)
*                           represents the Jth coefficient of the
*                           numerator of the (J,K)th term in the
*                           partial fraction expansion.  Namely,
*                           that term with the Jth factor of DEN
*                           to the Kth power as denominator.
*
*           DEGX            (input) integer
*                           is a two-dimensional array.  DEG(I,J)
*                           represents the degree of the numerator
*                           of the (I,J)th term in the partial
*                           fraction expansion.  See the description
*                           of X.
*
*       RETURN:             Not used.
*
*
*       ROUTINES
*       CALLED:             EUCLID,PUTX,GETD,GETX,POLADD
*
*
*       AUTHOR:             James F. Stafford
*
*
*       DATE CREATED:       8Jun87  Version 1.0
*
*
*       REVISIONS:          None.
*
*
******************************************************************
        SUBROUTINE          DIFFERENT(I,B,DEGB,DEN,DEGD,MULTS,X,DEGX)

        IMPLICIT            NONE

        INTEGER             DEGD(*),MULTS(*),I,J,K,L,DEGS,
     +                      DEGT,DEGX(10,*),DEGA,DEGB,DEGF

        REAL*8              DEN(0:2,*),X(0:1,10,*),A(0:2),B(0:2),
     +                      S(0:1),T(0:1),F(0:1)

        DO J=I-1,1,-1

            PRINT *,'J=',J

            CALL GETD(J,A,DEGA,DEN,DEGD)

            CALL GETX(J,MULTS(J),F,DEGF,X,DEGX)
```

```
      DO K=MULTS(J)-1,1,-1
         PRINT *,'K=',K
         CALL EUCLID(A,DEGA,B,DEGB,F,DEGF,S,DEGS,T,DEGT)
         CALL PUTX(J,K+1,T,DEGT,X,DEGX)
         CALL GETX(J,K,F,DEGF,X,DEGX)
         CALL POLADD(F,DEGF,S,DEGS)
      ENDDO
      CALL EUCLID(A,DEGA,B,DEGB,F,DEGF,S,DEGS,T,DEGT)
      CALL PUTX(J,1,T,DEGT,X,DEGX)
      CALL GETX(I,1,F,DEGF,X,DEGX)
      CALL POLADD(F,DEGF,S,DEGS)
      CALL PUTX(I,1,F,DEGF,X,DEGX)
   ENDDO
   RETURN
   END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:EUCLID.FOR                   *
*****************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        EUCLID(POL1,DEG1,POL2,DEG2,GCD,DEGG)
*
*
*      DESCRIPTION:      This program computes the greatest
*                        common divisor of two given polynomials.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:
*
*          POL1          (input) real
*                        is an array representing one input
*                        polynomial.
*
*          DEG1          (input) integer
*                        is the degree of POL1.
*
*          POL2          (input) real
*                        is an array representing the other
*                        input polynomial.
*
*          DEG2          (input) integer
*                        is the degree of POL2.
*
*          GCD           (output) real
*                        is the gcd of the two input polynomials
*
*          DEGG          (output) integer
*                        is the degree of GCD.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           POLDIV
*
*
*      AUTHOR:           James F. Stafford
```

```
*
*
*       DATE CREATED:    30Jun88 Version 1.0
*
*
*       REVISIONS:       None.
*
*
******************************************************************
        SUBROUTINE EUCLID(POL1,DEG1,POL2,DEG2,GCD,DEGG)

        IMPLICIT         NONE

        INTEGER          I,J,DEG1,DEG2,DEGG,DEGA,DEGB,DEGQ

        REAL*8           POL1(0:10),POL2(0:10),GCD(0:10),
     +                   A(0:10),B(0:10),Q(0:10),ZERO

        LOGICAL          EASY,HARD

        PARAMETER        (ZERO=1.0E-5)


        EASY=.FALSE.
        HARD=.TRUE.

        CALL GET(GCD,DEGG,POL1,DEG1)
        CALL GET(B,DEGB,POL2,DEG2)

        DO WHILE (.NOT.(DEGB.EQ.0.AND.DABS(B(0)).LT.ZERO))

            CALL GET(A,DEGA,GCD,DEGG)

            CALL GET(GCD,DEGG,B,DEGB)

            CALL POLDIV(A,DEGA,GCD,DEGG,Q,DEGQ,B,DEGB,EASY,HARD)

            DO I=0,DEGG

                GCD(I)=GCD(I)/GCD(DEGG)

            ENDDO

        ENDDO

        RETURN

        END

        SUBROUTINE       GET(A,DEGA,B,DEGB)
```

```
IMPLICIT        NONE
INTEGER         I,J,DEGA,DEGB
REAL*8          A(0:*),B(0:*)
DEGA=DEGB
DO I=0,DEGA
   A(I)=B(I)
ENDDO
RETURN
END
```

```
******************************************************************
*      Department of Electrical and Computer Engineering        *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:    EUCLIDEAN.FOR            *
******************************************************************
*
*      ROUTINE:           SUBROUTINE
*                         EUCLID(A,DEGA,B,DEGB,F,DEGF,S,DEGS,
*                         T,DEGT)
*
*      DESCRIPTION:       This program computes A and A^* in
*                         equation (4.1) given K, s and r via
*                         the methods discussed in chapter IV.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:         The following arguments are passed to
*                         the subroutine:
*
*          A              (input) real
*                         is an array containing the coefficients
*                         of s in (4.2).
*
*          DEGA           (input) integer
*                         is the degree of s in (4.2).
*
*          B              (input) real
*                         is an array containing the coefficients
*                         of r in (4.2).
*
*          DEGB           (input) integer
*                         is the degree of r in (4.2).
*
*          F              (input) real
*                         is an array containing the coefficients
*                         of K in (4.2).
*
*          DEGF           (input) integer
*                         is the degree of K in (4.2).
*
*          S              (output) real
*                         is an array containing the coefficients
*                         of A in (4.2).
*
*          DEGS           (output) integer
*                         is the degree of A in (4.2).
```

```
*
*          T              (output) real
*                         is an array containing the coefficients
*                         of A^* in (4.2).
*
*          DEGT           (output) integer
*                         is the degree of A^* in (4.2).
*
*      RETURN:            Not used.
*
*
*      ROUTINES
*      CALLED:            POLDIV,POLMULT,
*                         GET(contained in TRANSFER)
*
*
*      AUTHOR:            James F. Stafford
*
*
*      DATE CREATED:      9Jun87   Version 1.0
*
*
*      REVISIONS:         None.
*
*
*******************************************************************
      SUBROUTINE          EUCLID(A,DEGA,B,DEGB,F,DEGF,S,DEGS,
     +                    T,DEGT)

      IMPLICIT            NONE

      INTEGER             I,DEGA,DEGB,DEGF,DEGT,DEGS,DEGQ,DEGR,
     +                    DEG1,DEG2,ADD,MULT,DIV

      REAL*8              A(0:2),B(0:2),F(0:1),S(0:1),T(0:1),
     +                    QUO(0:1),REM(0:2),BUFF1(0:3),BUFF2(0:3)

      LOGICAL             EASYDIV,HARD,EASYMULT,SWITCH

      ADD=0
      DIV=0
      MULT=0
      EASYDIV=.TRUE.
      HARD=.FALSE.
      DEGR=1

      IF (DEGA.LE.DEGB)  THEN

         SWITCH=.TRUE.
```

```
          CALL GET(BUFF2,DEG2,A,DEGA)
          CALL GET(BUFF1,DEG1,B,DEGB)

     ELSE

          SWITCH=.FALSE.

          CALL GET(BUFF1,DEG1,A,DEGA)
          CALL GET(BUFF2,DEG2,B,DEGB)


     ENDIF

     I=0

     DO WHILE (DEGR.GT.0)

          CALL POLDIV(BUFF1,DEG1,BUFF2,DEG2,QUO,DEGQ,REM,DEGR,
+         EASYDIV,HARD)

          CALL GET(BUFF1,DEG1,BUFF2,DEG2)
          CALL GET(BUFF2,DEG2,REM,DEGR)

          EASYDIV=.FALSE.
          HARD=.TRUE.
          I=I+1

     ENDDO

     IF (I.LT.2) THEN

          EASYMULT=.TRUE.
          DEGQ=0

     ELSE

          EASYMULT=.FALSE.
          BUFF2(0)=-BUFF2(0)
          ADD=ADD+1

     ENDIF

     CALL POLMULT(QUO,DEGQ,F,DEGF,BUFF1,DEG1,EASYMULT)

     DO I=0,DEG1

          BUFF1(I)=BUFF1(I)/BUFF2(0)
          DIV=DIV+1

     ENDDO
```

```
      HARD=.FALSE.

      IF (SWITCH) THEN

          CALL POLDIV(BUFF1,DEG1,A,DEGA,QUO,DEGQ,T,DEGT,
     +        EASYDIV,HARD)

          DEGS=DEGB-1
          S(DEGS)=T(DEGT)
          DO I=DEGS-1,0,-1

              S(0)=T(DEGT)*(B(DEGB-1)-A(DEGA-1))
              MULT=MULT+1
              ADD=ADD+1

              IF (DEGT.GT.0) THEN

                  S(0)=S(0)+T(0)
                  ADD=ADD+1

              ENDIF

          ENDDO

      ELSE

          CALL POLDIV(BUFF1,DEG1,B,DEGB,QUO,DEGQ,S,DEGS,
     +        EASYDIV,HARD)

          DEGT=1
          S(0)=-S(0)
          T(1)=S(0)
          T(0)=(A(1)-B(0))*S(0)+F(1)
          ADD=ADD+3
          MULT=MULT+1

      ENDIF

      PRINT *,ADD,'additions'
      PRINT *,MULT,'multiplies'
      PRINT *,DIV,'divisions'

      RETURN

      END
```

```
*******************************************************************
*      Department of Electrical and Computer Engineering          *
*                  Kansas State University                        *
*                                                                 *
*      VAX FORTRAN source filename:    EXPAND.FOR                 *
*******************************************************************
*
*      ROUTINE:            SUBROUTINE
*                          EXPAND((NUM,DEGN,DEN,DEGD,MULTS,X,DEGX,
*                          NO_FACTS)
*
*
*      DESCRIPTION:        This program performs a partial fraction
*                          expansion on a rational function using
*                          Chin's and Steiglitz's algorithm.
*
*
*      DOCUMENTATION
*      FILES:              None.
*
*
*      ARGUMENTS:          The following arguments are passed to
*                          the subroutine:
*
*          NUM             (input) real
*                          is an array containing the coefficients
*                          of the numerator polynomial of the
*                          rational function to be expanded.
*
*          DEGN            (input) integer
*                          is the degree of the numerator
*                          polynomial.
*
*          DEN             (input) real
*                          is a two-dimensional array.  DEN(I,J)
*                          represents the coefficient of the Ith
*                          power of x in the Jth factor of the
*                          denominator polynomial.
*
*          DEGD            (input) integer
*                          is an array.  DEGD(I) represents the
*                          degree of the Ith factor in the
*                          denominator polynomial.
*
*          MULTS           (input) integer
*                          is an array.  MULTS(I) represents the
*                          multiplicity of the Ith factor in the
*                          denominator polynomial.
*
*          NO_FACTS        (input) integer
```

```
*                           is the number of factors in the
*                           denominator polynomial.
*
*         X                 (output) real
*                           is a three-dimensional array.  X(I,J,K)
*                           represents the Ith coefficient of the
*                           numerator of the (J,K)th term in the
*                           partial fraction expansion.  Namely,
*                           that term with the Jth factor of DEN
*                           to the Kth power as denominator.
*
*         DEGX              (output) integer
*                           is an array.  DEGX(I,J) represents the
*                           degree of the numerator of the (I,J)th
*                           term in the partial fraction expansion.
*                           See the description of X.
*
*
*      RETURN:              Not used.
*
*
*      ROUTINES
*      CALLED:              GETD,POLDIV,ALIKE,DIFFERENT
*
*
*      AUTHOR:              James F. Stafford
*
*
*      DATE CREATED:        6Jun87   Version 1.0
*
*
*      REVISIONS:           None.
*
*
*****************************************************************
       SUBROUTINE           EXPAND(NUM,DEGN,DEN,DEGD,MULTS,X,DEGX,
     +                      NO_FACTS)

       IMPLICIT             NONE

       INTEGER              DEGN,NO_FACTS,DEGD(10),DEGB,
     +                      MULTS(*),DEGR,I,DEGQ,J,K,L,
     +                      DEGX(10,*)

       REAL*8               NUM(0:*),DEN(0:2,*),X(0:1,10,*),
     +                      QUO(0:10),REM(0:10),B(0:2)

       LOGICAL              EASY,HARD

       EASY=.FALSE.
```

```
      HARD=.FALSE.

      DO I=1,NO_FACTS

          PRINT *,'I=',I

          CALL GETD(I,B,DEGB,DEN,DEGD)

          DO J=1,MULTS(I)

              CALL POLDIV(NUM,DEGN,B,DEGB,QUO,DEGQ,REM,DEGR,
     +        EASY,HARD)

              DEGN=DEGQ

              DO K=0,DEGN

                  NUM(K)=QUO(K)

              ENDDO

              PRINT *,DEGR,'YES'

              DO K=0,DEGR

                  PRINT *,REM(K)

              ENDDO

              CALL ALIKE(I,J,X,DEGX,REM,DEGR)

              CALL DIFFERENT(I,B,DEGB,DEN,DEGD,MULTS,X,DEGX)

          ENDDO

      ENDDO

      RETURN

      END
```

```
*******************************************************************
*       Department of Electrical and Computer Engineering         *
*                    Kansas State University                      *
*                                                                 *
*       VAX FORTRAN source filename:FACTORER.FOR                  *
*******************************************************************
*
*       ROUTINE:            SUBROUTINE
*                           FACTORER(POLY,DEGP,FACTOR,NUM,DEGF,MULT)
*
*
*       DESCRIPTION:        This program factors a given input
*                           polynomial into irreducible elements of
*                           R[x].
*
*
*       DOCUMENTATION
*       FILES:              None.
*
*
*       ARGUMENTS:
*
*           POLY            (input) real
*                           is an array containing the coefficients
*                           the polynomial to be factored.
*
*           DEGP            (input) integer
*                           is the degree of POLY.
*
*           FACTOR          (output) real
*                           is an array containinng coefficients
*                           each factor of POLY.
*
*           NUM             (output) integer
*                           is the number of factors in FACTOR.
*
*           DEGF            (output) integer
*                           is an array specifying the degree of
*                           the corresponding factor in FACTOR.
*
*           MULT            (output) integer
*                           is an array specifying the multiplicity
*                           of the corresponding factor in FACTOR.
*
*       RETURN:             Not used.
*
*
*       ROUTINES
*       CALLED:             DERIV, EUCLID, FIND_EM
*
```

```
*
*       AUTHOR:           James F. Stafford
*
*
*       DATE CREATED:     30Jun88 Version 1.0
*
*
*       REVISIONS:        None.
*
*
*****************************************************************
        SUBROUTINE        FACTORER(POLY,DEGP,FACTOR,NUM,DEGF,MULT)

        IMPLICIT          NONE

        INTEGER           I,J,K,DEGP,DEGF(*),MULT(*),NUM,DEGGCD(0:10),
     +                    DEGD,DEGG

        REAL*8            POLY(0:10),FACTOR(10,0:2),GCD(0:10,0:10),D(0:10)
     +                    G(0:10)

        DO I=0,DEGP

            D(I)=POLY(I)
            GCD(0,I)=POLY(I)

        ENDDO

        DEGD=DEGP
        DEGGCD(0)=DEGP
        K=0

        DO WHILE (DEGGCD(K).GT.0)

            K=K+1
            CALL DERIV(D,DEGD)
            CALL EUCLID(POLY,DEGP,D,DEGD,G,DEGG)

            DO J=0,DEGG

                GCD(K,J)=G(J)

            ENDDO

            DEGGCD(K)=DEGG

        ENDDO

        DO I=K-1,0,-1
```

```
        DO J=0,DEGGCD(I)
            G(J)=GCD(I,J)
        ENDDO
        DEGG=DEGGCD(I)
        CALL FIND_EM(G,DEGG,FACTOR,NUM,DEGF,MULT)
    ENDDO
    RETURN
    END
```

```
*******************************************************************
*      Department of Electrical and Computer Engineering         *
*                 Kansas State University                         *
*                                                                 *
*      VAX FORTRAN source filename:FIND_EM.FOR                    *
*******************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        FIND_EM(P,DEGP,FACTOR,NUM,DEGF,MULT)
*
*
*      DESCRIPTION:      A polynomial and a list of already known
*                        roots is passed to the subroutine.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           MULLER
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     5Sep86  Version 1.0
*
*
*      REVISIONS:        None.
*
*
*******************************************************************
       SUBROUTINE        FIND_EM(P,DEGP,FACTOR,NUM,DEGF,MULT)

       IMPLICIT          NONE

       INTEGER           J,DEGP,NUM,DEGF(*),MULT(*),SUM

       REAL*8            P(0:*),FACTOR(10,0:2)

       SUM=0

       DO J=1,NUM
```

```
    MULT(J)=MULT(J)+1
    SUM=SUM+DEGF(J)*MULT(J)

ENDDO

DO WHILE (DEGP-SUM.GT.0)

    CALL MULLER(P,DEGP,FACTOR,NUM,DEGF,MULT)
    SUM=SUM+DEGF(NUM)

ENDDO

RETURN

END
```

```
*********************************************************************
*     Department of Electrical and Computer Engineering            *
*                    Kansas State University                       *
*                                                                  *
*     VAX FORTRAN source filename:    GAMMA.FOR                    *
*********************************************************************
*
*     ROUTINE:           FUNCTION
*                        GAMMA(K)
*
*
*     DESCRIPTION:       This program computes the integer-valued
*                        function Gamma defined in the glossary.
*
*     DOCUMENTATION
*     FILES:             None.
*
*
*     ARGUMENTS:         The following argument is passed to the
*                        function:
*
*        K               (input) integer
*                        is any non-negative integer.
*
*     RETURN:            The function returns an integer according
*                        to the definition in the glossary.
*
*
*     ROUTINES
*     CALLED:            None.
*
*
*     AUTHOR:            James F. Stafford
*
*
*     DATE CREATED:      5Sep86   Version 1.0
*
*
*     REVISIONS:         None.
*
*
********************************************************************
      INTEGER FUNCTION        GAMMA(K)

      IMPLICIT         NONE

      INTEGER          J,K

      GAMMA=1
```

```
DO J=K-1,2,-1
    GAMMA=GAMMA*J
ENDDO
RETURN
END
```

```
******************************************************************
*      Department of Electrical and Computer Engineering        *
*                   Kansas State University                      *
*                                                                *
*      VAX FORTRAN source filename:    INIT.FOR                  *
******************************************************************
*
*      ROUTINE:           SUBROUTINE
*                         INIT(F)
*
*      DESCRIPTION:       This program initializes the recursively
*                         defined function H_k described in
*                         chapter V.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:         The following argument in passed to
*                         the subroutine:
*
*         F               (output) real
*                         is an array containing the coefficients
*                         of the functions H_k.  For a given j,
*                         F(I,0,K) represents the coefficient of
*                         the cosine term of H_(j-I), with t to the
*                         power K.  F(I,1,K) represents the
*                         coefficient of the sine term of H_(j-I)
*                         with t to the power K.
*
*      RETURN:            Not used.
*
*
*      ROUTINES
*      CALLED:            None.
*
*
*      AUTHOR:            James F. Stafford
*
*
*      DATE CREATED:      9Jun87  Version 1.0
*
*
*      REVISIONS:         None.
*
*
******************************************************************
       SUBROUTINE      INIT(F)
```

```
IMPLICIT        NONE

INTEGER         I,J,K

REAL*8          F(-2:0,0:1,-1:9)

DO I=-2,0

   DO J=0,1

      DO K=-1,9

         F(I,J,K)=0.

      ENDDO

   ENDDO

ENDDO

F(-1,0,-1)=1.
F(0,1,0)=1.

RETURN

END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                     *
*                                                              *
*      VAX FORTRAN source filename:    INPUT_RAT.FOR           *
*****************************************************************
*
*      ROUTINE:          PROGRAM
*
*
*      DESCRIPTION:      This program allows one to establish a
*                        data file compatible with the inverse
*                        transform package programs containing the
*                        necessary data to describe a rational
*                        function.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        Not used.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           None.
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     27May87 Version 1.0
*
*
*      REVISIONS:        None.
*
*
*****************************************************************
       IMPLICIT          NONE

       INTEGER           I,N_DEG,D_DEG,NO_ROOTS,MLTPLCTS(10)

       REAL*8            NUM(0:15),DEN(0:15)

       COMPLEX*16        ROOTS(10)

       CHARACTER*15      FILENAME,YESNO
```

```
      PRINT *,'Are numerator roots known? (Y/N)'
      READ (*,200) YESNO
      IF (YESNO.EQ.'Y') THEN
          CALL INPUT_FACT(NO_ROOTS,ROOTS,MLTPLCTS)
          CALL RECONSTRUCT(NO_ROOTS,ROOTS,MLTPLCTS,
     +        NUM,N_DEG)
      ELSE
          CALL INPUT_NONFACT(NUM,N_DEG)
      ENDIF
      PRINT *,'Are denominator roots known? (Y/N)'
      READ (*,200) YESNO
      IF (YESNO.EQ.'Y') THEN
          CALL INPUT_FACT(NO_ROOTS,ROOTS,MLTPLCTS)
          CALL RECONSTRUCT(NO_ROOTS,ROOTS,MLTPLCTS,
     +        DEN,D_DEG)
      ELSE
          CALL INPUT_NONFACT(DEN,D_DEG)
      ENDIF
      PRINT *,'Enter filename.'
      READ (*,200) FILENAME
200   FORMAT (A15)
      OPEN (UNIT=1, FILE=FILENAME, STATUS='NEW')
      WRITE (1,*) N_DEG
      DO I=0,N_DEG
          WRITE (1,*) NUM(I)
      ENDDO
```

```
      WRITE (1,*) D_DEG

      DO I=0,D_DEG

          WRITE (1,*) DEN(I)

      ENDDO

      CLOSE (UNIT=1, STATUS='KEEP')

      END

      SUBROUTINE       INPUT_FACT(NO_ROOTS,ROOTS,MLTPLCTS)

      IMPLICIT         NONE

      INTEGER          NO_ROOTS,MLTPLCTS(*),I

      COMPLEX*16       ROOTS(*)

      PRINT *,'Input number of roots'

      READ (*,*) NO_ROOTS

100   FORMAT (F8.5,F8.5)

      DO I=1,NO_ROOTS

          PRINT *,'Input root number ',I
          READ (*,100) ROOTS(I)
          PRINT *,'Input corresponding multiplicity'
          READ (*,*) MLTPLCTS(I)

          PRINT *,ROOTS(I),MLTPLCTS(I)

      ENDDO

      RETURN

      END

      SUBROUTINE       RECONSTRUCT(NO_ROOTS,ROOTS,MLTPLCTS,
     +                 RESULT,ORDER)

      IMPLICIT         NONE

      INTEGER          NO_ROOTS,MLTPLCTS(*),I,J,ORDER,MULT

      COMPLEX*16       ROOTS(*),BUFFER(0:50)
```

```
REAL*8              RESULT(0:*)

BUFFER(0)=DCMPLX(1.0,0.0)

DO I=1,50

    BUFFER(I)=DCMPLX(0.,0.)

ENDDO

ORDER=0

DO I=1,NO_ROOTS

    ORDER=ORDER+MLTPLCTS(I)
    MULT=MLTPLCTS(I)

    DO WHILE (MULT.NE.0)

        DO J=ORDER,1,-1

            BUFFER(J)=BUFFER(J-1)-BUFFER(J)*ROOTS(I)

        ENDDO

        BUFFER(0)=-BUFFER(0)*ROOTS(I)
        MULT=MULT-1

    ENDDO

ENDDO

DO I=0,ORDER

    RESULT(I)=REAL(BUFFER(I))

ENDDO

RETURN

END
SUBROUTINE        INPUT_NONFACT(POLY,DEG)

IMPLICIT          NONE

INTEGER           DEG,I

REAL*8            POLY(0:*)
```

```
PRINT *,'Input degree'

READ (*,*) DEG

DO I=0,DEG

    PRINT *,'Input coeff. of power ',I
    READ (*,*) POLY(I)

ENDDO

RETURN

END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering        *
*                    Kansas State University                    *
*                                                               *
*      VAX FORTRAN source filename:    INV2.FOR                 *
*****************************************************************
*
*      ROUTINE:            SUBROUTINE
*                          INV2(I,J,RESP,OMEGA,A,B)
*
*      DESCRIPTION:        This program computes the inverse
*                          Laplace transform of a rational function
*                          such that equation (5.2) applies.  Note
*                          that since the algorithm described in
*                          chapter V is recursive, that for a
*                          given k, all of the previous transforms
*                          for k>j>=1 must be already computed.
*                          The function H_k is computed each time
*                          the subroutine is called, using H_(k-1)
*                          and H_(k-2) which are held in an array
*                          intrinsic to this routine, namely, F.
*                          On K=1, F is initialized to hold the
*                          coefficients of H_0 and H_-1. The
*                          inverse transform coefficients are
*                          accumulated in an array called RESP.
*
*      DOCUMENTATION
*      FILES:              None.
*
*
*      ARGUMENTS:          The following arguments are passed to
*                          the subroutine:
*
*          I               (input) integer
*                          is an index variable specifying which
*                          factor of the denominator polynomial
*                          of the original rational function is
*                          currently being inverse transformed.
*
*          J               (input) integer
*                          corresponds to k in (5.2)
*
*          RESP            (input/output) real\
*                          is an array to accumulate the computed
*                          time-domain response.  RESP(j,0,i)
*                          represents alpha_ji in equation (5.5)
*                          and RESP(j,1,i) respresents beta_ji
*                          in equation (5.5).
*
*          OMEGA           (input) real
```

```
*                             corresponds to a in (5.2).
*
*          A                  (input) real
*                             corresponds to A in (5.2).
*
*          B                  (input) real
*                             corresponds to B in (5.2).
*
*      RETURN:                Not used.
*
*
*      ROUTINES
*      CALLED:                INIT,BESSEL,GAMMA
*
*
*      AUTHOR:                James F. Stafford
*
*
*      DATE CREATED:          5Sep86   Version 1.0
*
*
*      REVISIONS:             None.
*
*
*****************************************************************
       SUBROUTINE             INV2(I,J,RESP,OMEGA,A,B)

       IMPLICIT               NONE

       INTEGER                I,J,K,L,M,GAMMA

       REAL*8                 A,B,OMEGA,F(-2:0,0:1,-1:9),RESP(10,0:1,0:9),
     +                        ADJ

       IF (J.EQ.1)  THEN

           CALL INIT(F)

       ELSE

           CALL BESSEL(F,OMEGA,J)

       ENDIF

       ADJ=GAMMA(J)*(2*OMEGA)**(J-1)

       DO K=0,J-1

           RESP(I,0,K)=RESP(I,0,K)+(F(-1,0,K-1)*A+F(0,0,K)*B)
     +     /ADJ
```

```
         RESP(I,1,K)=RESP(I,1,K)+(F(-1,1,K-1)*A+F(0,1,K)*B)
+        /ADJ

      ENDDO

      RETURN

      END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering        *
*                 Kansas State University                       *
*                                                               *
*      VAX FORTRAN source filename:    INVERT.FOR               *
*****************************************************************
*
*      ROUTINE:          PROGRAM
*                        INVERT
*
*      DESCRIPTION:      This program computes the inverse
*                        Laplace transform of a rational
*                        function that has been expanded into
*                        partial fractions.  The user is prompted
*                        for a filename under which the
*                        output of the partial fraction expander
*                        program has been stored.  Again, the
*                        user is prompted for another filename
*                        under which to store the parameters of
*                        the inverse transform function.
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        None.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           INV2
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     10Jun87 Version 1.0
*
*
*      REVISIONS:        None.
*
*
*****************************************************************
       PROGRAM           INVERT

       IMPLICIT          NONE

       INTEGER           I,J,K,NO_TERMS,ORDER,MULT(10),GAMMA
```

```
      REAL*8          TAU(10),OMEGA(10),A,B,F(-2:0,0:1,-1:10),
     +                RESP(10,0:1,0:9)

      CHARACTER*15    FILENAME

      PRINT *,'Enter filename.'

      READ (*,200) FILENAME

200   FORMAT (A15)

      OPEN (UNIT=1, FILE=FILENAME, STATUS='OLD')

      READ (1,*) NO_TERMS

      DO I=1,NO_TERMS

          READ (1,*) ORDER
          PRINT *,ORDER
          READ (1,*) MULT(I)
          PRINT *,MULT(I)

          IF (ORDER.EQ.1) THEN

              READ (1,*) TAU(I)
              PRINT *,TAU(I)

          ELSE

              READ (1,*) TAU(I)
              PRINT *,TAU(I)
              READ (1,*) OMEGA(I)
              PRINT *,OMEGA(I)

          ENDIF

          DO J=1,MULT(I)

              IF (ORDER.EQ.1) THEN

                  READ (1,*) A
                  PRINT *,A
                  RESP(I,0,J-1)=A/GAMMA(J)

              ELSE

                  READ (1,*) A
                  PRINT *,'A =',A
                  READ (1,*) B
```

```
                PRINT *,'B =',B

                CALL INV2(I,J,RESP,OMEGA(I),A,B)

             ENDIF

          ENDDO

       ENDDO

       CLOSE (UNIT=1, STATUS='KEEP')

       PRINT *,'Enter filename.'

       READ (*,200) FILENAME

       OPEN (UNIT=1, FILE=FILENAME, STATUS='NEW')

       WRITE (1,*) NO_TERMS

       DO I=1,NO_TERMS

             WRITE(*,300) 'exp(',-TAU(I),'t)*'
             WRITE(1,*) TAU(I),OMEGA(I)
             WRITE(1,*) MULT(I)

          DO J=1,MULT(I)

                WRITE(*,301) '(',RESP(I,0,J-1),'t^',J-1,
     +          'COS',OMEGA(I),'t + '
                WRITE(1,*) RESP(I,0,J-1)
                WRITE(*,301) ' ',RESP(I,1,J-1),'t^',J-1,
     +          'SIN',OMEGA(I),'t)  '
                WRITE(1,*) RESP(I,1,J-1)

          ENDDO

       ENDDO

       CLOSE (UNIT=1, STATUS='KEEP')

300    FORMAT  (A5,E12.4E3,A3)
301    FORMAT  (A2,E12.4E3,A2,I2,A3,E12.4E3,A4)

       END
```

```
*******************************************************************
*      Department of Electrical and Computer Engineering          *
*                  Kansas State University                        *
*                                                                 *
*      VAX FORTRAN source filename:  LOTS_O_PLOTS.FOR             *
*******************************************************************
*
*   ROUTINE:              subroutine
*                         FIRST_PLOT(DEVICE,NUM_POINTS,X_DATA,
*                         Y_DATA,X_AXIS_TITLE,X_AXIS_UNITS,
*                         Y_AXIS_TITLE,Y_AXIS_UNITS,PLOT_TITLE,
*                         INFO)
*
*
*      DESCRIPTION:       Makes a plot using X_DATA as abscissa
*                         and Y_DATA as ordinate.  The axes are
*                         labelled with titles and units.  The
*                         plot is also titled.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:
*
*          DEVICE         (input)  integer
*                         is the device type to display the plot
*
*                         7475     for plotter
*                         4014     for terminal (Selanar only)
*
*          NUM_POINTS     (input)  integer
*                         is the number of data points to
*                         be plotted
*
*          X_DATA         (input)  real
*                         is the array of abscissa values for the
*                         data to be plotted
*
*          Y_DATA         (input)  real
*                         is the array of ordinate values for the
*                         data to be plotted
*
*          X_AXIS_TITLE (input)  character*(*)
*                         is the title to be placed on the x_axis
*
*          X_AXIS_UNITS (input)  character*(*)
*                         is the name to be given to the units
```

```
                          associated with the x-axis
*
*          Y_AXIS_TITLE (input)  character*(*)
*                       is the title to be placed on the y-axis
*
*          Y_AXIS_UNITS (input)  character*(*)
*                       is the name to be given to the units
*                       associated with the y-axis
*
*          PLOT_TITLE   (input)  character*(*)
*                       is the title to be placed on the plot
*
*      RETURN:
*
*          INFO        (output)  real(6)
*                      is the information necessary to make
*                      subsequent plots on the same axes.
*
*
*      ROUTINES
*      CALLED:         P System of Generalized Plot Routines
*
*
*      AUTHOR:         James F. Stafford
*
*
*      DATE CREATED:   24May86  Version 1.0
*
*
*      REVISIONS:      None.
*
*
**************************************************************

      SUBROUTINE  FIRST_PLOT(DEVICE,NUM_POINTS,X_DATA,Y_DATA,
     +                X_AXIS_TITLE,X_AXIS_UNITS,Y_AXIS_TITLE,
     +               Y_AXIS_UNITS,PLOT_TITLE,INFO)

      IMPLICIT NONE

      INTEGER  DEVICE,NUM_POINTS,FORLAB,FORTIC,NEGFLG,FORM,
     +         SCNTL,LENSTR,UPDOWN

      REAL     X_DATA(*),Y_DATA(*),FACTOR,VEL,X,Y,LENGTH,
     +         FIRSTX,DELTAX,ANGLE,CLEN,FIRDEL(4),
     +         DIVLNX,DIVLNY,WIDTH,HEIGHT,INFO(6)

      CHARACTER*(*)  X_AXIS_TITLE,Y_AXIS_TITLE,X_AXIS_UNITS,
     +               Y_AXIS_UNITS,PLOT_TITLE
```

```
      CHARACTER*(1)  BLANK,SIZE

*INTIALIZE PLOT DEVICE

      FACTOR=1.0
      BLANK=' '
      SIZE='A'

      CALL  PINIT(DEVICE,BLANK,FACTOR,SIZE)

*SET PEN VELOCITY

      VEL=10.0

      CALL  PSTVEL(VEL)

*ESTABLISH ORIGIN

      X=4.5
      Y=4.5

      CALL PORIG(X,Y)

*SET OFFSETS FOR AXIS ROUTINES (RELATIVE TO ORIGIN)

      X=0.0
      Y=0.0

*DRAW Y-AXIS AND LABEL

          LENGTH=12.0

          CALL PSCALE(Y_DATA,NUM_POINTS,LENGTH,FIRSTX,
     +    DELTAX,DIVLNY)

          FIRDEL(3)=FIRSTX
          FIRDEL(4)=DELTAX
          FORLAB=110
          FORTIC=1001
          ANGLE=90.0

          CALL  PAXIS(X,Y,Y_AXIS_TITLE,Y_AXIS_UNITS,FORLAB,
     +    FORTIC,LENGTH,ANGLE,FIRSTX,DELTAX,DIVLNY)

*DRAW X-AXIS AND LABEL

          LENGTH=18

          CALL PSCALE(X_DATA,NUM_POINTS,LENGTH,FIRSTX,
```

```
     +      DELTAX,DIVLNX)

            FIRDEL(1)=FIRSTX
            FIRDEL(2)=DELTAX
            FORLAB=211
            FORTIC=2001
            ANGLE=0.0

            CALL PAXIS(X,Y,X_AXIS_TITLE,X_AXIS_UNITS,FORLAB,
     +      FORTIC,LENGTH,ANGLE,FIRSTX,DELTAX,DIVLNX)

*DRAW CURVE

            SCNTL=0

            CALL PLINE(X_DATA,Y_DATA,NUM_POINTS,FIRDEL,SCNTL,
     +      BLANK,DIVLNX,DIVLNY)

*TITLE THE PLOT

        UPDOWN=0
        X=9.0
        Y=13.0

        INFO(1)=FIRDEL(1)
        INFO(2)=FIRDEL(2)
        INFO(3)=FIRDEL(3)
        INFO(4)=FIRDEL(4)
        INFO(5)=DIVLNX
        INFO(6)=DIVLNY

        CALL PPLOT(X,Y,UPDOWN)

        CALL PTXTLN(PLOT_TITLE,LENSTR)

        WIDTH=-LENSTR/2
        HEIGHT=0.0

        CALL PCHRPL(WIDTH,HEIGHT)

        CALL PTEXT(PLOT_TITLE)

*       CALL PCLOSP

        RETURN

        END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                     *
*                                                              *
*      VAX FORTRAN source filename:    MULLER.FOR              *
*****************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        MULLER(POLY,DEG,X)
*
*
*      DESCRIPTION:      This program uses Muller's method
*                        (page 262, Numerical Recipes) to
*                        find a root of a polynomial.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        The following arguments are passed
*                        to the subroutine:
*
*          POLY          (input) real
*                        is an array containing the coefficients
*                        of the polynomial of interest.
*
*          DEG           (input) integer
*                        is the degree of the above polynomial.
*
*          FACTOR        (input/output)
*                        is an array containing known roots of
*                        the polynomial represented by POLY.
*
*          NUM           (input/output)
*                        is the number of factors in FACTOR.
*
*          DEGF          (input/output)
*                        is an array containing the degree of
*                        each corresponding factor in FACTOR.
*
*          MULT          (input/output)
*                        is an array containing the multiplicity
*                        of each corresponding factor in FACTOR
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           DEFVAL, COMPOSE
```

```
*
*
*      AUTHOR:            James F. Stafford
*
*
*      DATE CREATED:      28May87 Version 1.0
*
*
*      REVISIONS:         30Jun88 Added deflation and factor table
*                         updating.
*
*
*****************************************************************
       SUBROUTINE         MULLER(POLY,DEG,FACTOR,NUM,DEGF,MULT)

       IMPLICIT           NONE

       INTEGER            DEG,NUM,DEGF(*),MULT(*),I,NO_ITERATIONS,MAX

       REAL*8             POLY(0:*),ZERO,FACTOR(10,0:2)

       COMPLEX*16         X(-2:1),Q,A,B,C,D,P(-2:0),DEFVAL

       PARAMETER          (ZERO=1.0E-12)

       PARAMETER          (MAX=200)

       NO_ITERATIONS=0

       X(-2)=DCMPLX(1.,1.)
       X(-1)=DCMPLX(1.,0.)
       X(0)=DCMPLX(1.,-1.)

       DO WHILE ((CDABS(X(0)-X(-1)).GT.CDABS(X(0))*ZERO)
     +          .AND.(CDABS(X(0)-X(-2)).GT.CDABS(X(0))*ZERO)
     +          .AND.(NO_ITERATIONS.LT.MAX))

          NO_ITERATIONS=NO_ITERATIONS+1
          B=DCMPLX(0.,0.)
          D=DCMPLX(0.,0.)

          DO WHILE ((D.EQ.DCMPLX(0.,0.)).AND.(B.EQ.DCMPLX(0.,0.)))

             DO I=-2,0

                P(I)=DEFVAL(POLY,DEG,FACTOR,NUM,DEGF,MULT,X(I))

             ENDDO

             Q=(X(0)-X(-1))/(X(-1)-X(-2))
```

```
          A=Q*P(0)-Q*(1+Q)*P(-1)+Q*Q*P(-2)
          B=(2*Q+1)*P(0)-((1+Q)**2)*P(-1)+Q*Q*P(-2)
          C=(1+Q)*P(0)
          D=SQRT(B*B-4*A*C)

          IF ((D.EQ.DCMPLX(0.,0.)).AND.(B.EQ.DCMPLX(0.,0.))) THEN

             X(-1)=(X(0)+X(-1))/2.
             X(-2)=(X(0)+X(-2))/2.

          ENDIF

       ENDDO

       IF (CDABS(B+D).GT.CDABS(B-D)) THEN

          X(1)=X(0)-(X(0)-X(-1))*2*C/(B+D)

       ELSE

          X(1)=X(0)-(X(0)-X(-1))*2*C/(B-D)

       ENDIF

       DO I=-2,0

          X(I)=X(I+1)

       ENDDO

    ENDDO

    PRINT *,'I MADE IT HERE',NO_ITERATIONS

    NUM=NUM+1
    CALL COMPOSE(X(1),FACTOR,NUM,DEGF,MULT)

    RETURN

    END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:    PART_FRAC.FOR            *
*****************************************************************
*
*      ROUTINE:          PROGRAM
*                        TEST
*
*
*      DESCRIPTION:      This program computes the partial
*                        fraction expansion of a rational
*                        function using the method described in
*                        the thesis.  The user is prompted
*                        for a filename under which the factored
*                        form of the rational function has been
*                        stored.  The user is prompted again
*                        for a filename under which to store the
*                        partial fraction expansion.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        None.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           SPEC_READ,PART_WRITE,EXPAND
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     10Jun87 Version 1.0
*
*
*      REVISIONS:        None.
*
*
*****************************************************************
       PROGRAM           TEST

       IMPLICIT          NONE
```

```
      INTEGER         I,J,K,DEGN,DEGD(10),NO_FACTS,MULTS(10),
     +                DEGX(10,5)
      REAL*8          NUM(0:15),DEN(0:2,10),X(0:1,10,5),
     +                FACT(0:2)

      LOGICAL         EASY,HARD

      CALL SPEC_READ(NUM,DEGN,DEN,DEGD,MULTS,NO_FACTS)

      CALL EXPAND(NUM,DEGN,DEN,DEGD,MULTS,X,DEGX,NO_FACTS)

      DO I=1,NO_FACTS

          DO J=1,MULTS(I)

              PRINT *,I,J

              DO K=0,DEGX(I,J)

                  PRINT *,X(K,I,J)

              ENDDO

          ENDDO

      ENDDO

      CALL PART_WRITE(NO_FACTS,MULTS,X,DEGX,DEN,DEGD)

      END
```

```
*******************************************************************
*       Department of Electrical and Computer Engineering         *
*                    Kansas State University                      *
*                                                                 *
*       VAX FORTRAN source filename:    PART_WRITE.FOR            *
*******************************************************************
*
*       ROUTINE:            SUBROUTINE
*                           PART_WRITE(NO_FACTS,MULTS,X,DEGX,DEN,
*                           DEGD)
*
*       DESCRIPTION:        This program writes the partial fraction
*                           expansion into a file.  The user is
*                           prompted for a filename.
*
*       DOCUMENTATION
*       FILES:              None.
*
*
*       ARGUMENTS:          The following arguments are passed to
*                           the subroutine:
*
*           NO_FACTS        (input) integer
*                           is the number of factors in the
*                           denominator polynomial.
*
*           MULTS           (input) integer
*                           is an array containing the
*                           multiplicities of each factor in
*                           the denominator polynomial.
*
*           X               (input) real
*                           is a three-dimensional array.  X(I,J,K)
*                           represents the Ith coefficient of the
*                           numerator of the (J,K)th term in the
*                           partial fraction expansion.  Namely,
*                           that term with the Jth factor of DEN
*                           to the Kth power as denominator.
*
*           DEGX            (input) integer
*                           is an array.  DEGX(I,J) represents the
*                           degree of the numerator of the (I,J)th
*                           term in the partial fraction expansion.
*                           See the description of X.
*
*           DEN             (input) real
*                           is a two-dimensional array.  DEN(I,J)
*                           represents the coefficient of the Ith
*                           power of x in the Jth factor of the
*                           denominator polynomial.
```

```
*
*          DEGD         (input) integer
*                       is an array.  DEGD(I) represents the
*                       degree of the Ith factor of the
*                       denominator polynomial.
*
*       RETURN:         Not used.
*
*
*       ROUTINES
*       CALLED:         None.
*
*
*       AUTHOR:         James F. Stafford
*
*
*       DATE CREATED:   7Jun87  Version 1.0
*
*
*       REVISIONS:      None.
*
*
*****************************************************************
        SUBROUTINE      PART_WRITE(NO_FACTS,MULTS,X,DEGX,DEN,DEGD)

        IMPLICIT        NONE

        INTEGER         I,J,K,DEGD(10),NO_FACTS,MULTS(10),
     +                  DEGX(10,5)

        REAL*8          DEN(0:2,10),X(0:1,10,5),ALPHA,BETA,A,B

        CHARACTER*15    FILENAME

        PRINT *,'Enter filename.'

        READ (*,200) FILENAME

200     FORMAT (A15)

        OPEN (UNIT=1, FILE=FILENAME, STATUS='NEW')

        WRITE (1,*) NO_FACTS

        DO I=1,NO_FACTS

           WRITE (1,*) DEGD(I)
           PRINT *,'ORDER =',DEGD(I)
           WRITE (1,*) MULTS(I)
           PRINT *,'MULTIPLICITY =',MULTS(I)
```

```
          IF (DEGD(I).EQ.1) THEN

              WRITE (1,*) DEN(0,I)
              PRINT *,'ALPHA =',DEN(0,I)

          ELSE

              ALPHA=DEN(1,I)/2
              BETA=DSQRT(DEN(0,I)-ALPHA**2)
              WRITE (1,*) ALPHA
              PRINT *,'ALPHA =',ALPHA
              WRITE (1,*) BETA
              PRINT *,'BETA =',BETA

          ENDIF

          DO J=1,MULTS(I)

              IF (DEGD(I).EQ.1) THEN

                  WRITE (1,*) X(0,I,J)
                  PRINT *,'A =',X(0,I,J)

              ELSE

                  A=X(1,I,J)
                  B=(X(0,I,J)-A*ALPHA)/BETA
                  WRITE (1,*) A
                  PRINT *,'A =',A
                  WRITE (1,*) B
                  PRINT *,'B =',B

              ENDIF

          ENDDO

      ENDDO

      CLOSE (UNIT=1, STATUS='KEEP')

      RETURN

      END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                   Kansas State University                     *
*                                                               *
*      VAX FORTRAN source filename:    PLOT.FOR                 *
*****************************************************************
*
*      ROUTINE:            PROGRAM
*                          PLOT
*
*      DESCRIPTION:        This program makes plots of time
*                          domain response functions computed
*                          by the inverse transform program.
*
*      DOCUMENTATION
*      FILES:              None.
*
*
*      ARGUMENTS:          None
*
*
*      RETURN:             Not used.
*
*
*      ROUTINES
*      CALLED:             FIRST_PLOT,READ,PLOT,
*                          PCLOSP (contained in the P System
*                          of Generalized Plotting Routines)
*
*      AUTHOR:             James F. Stafford
*
*
*      DATE CREATED:       24May87 Version 1.0
*
*
*      REVISIONS:          None.
*
*
*****************************************************************
       PROGRAM             TEST

       IMPLICIT            NONE

       INTEGER             DEVICE,NUM_POINTS,I,NUM_FILES

       REAL                X_DATA(1000),Y_DATA(1000),
     +                     INFO(6),TONE,TTWO

       CHARACTER*(15)      X_TITLE,Y_TITLE,X_UNITS,Y_UNITS,
     +                     TITLE,FILES(5)
```

```
      PRINT *,'INPUT <7475> FOR PLOTTER OR <4014> FOR TERMINAL'
      READ (*,*)DEVICE
      NUM_POINTS=1000
      X_TITLE='TIME'
      Y_TITLE='VALUE'
      X_UNITS='INTERVALS'
      Y_UNITS='UNITS'
      TITLE='TEST PLOT'

      PRINT *,'Input number of files to plot'
      READ (*,*) NUM_FILES

      DO I=1,NUM_FILES

          PRINT *,'Input name of file number ',I
          READ (*,200) FILES(I)

      ENDDO

200   FORMAT (A15)

      PRINT *,'Input initial time'
      READ (*,*) TONE

      PRINT *,'Input final time'
      READ (*,*) TTWO

      DO I=1,NUM_POINTS

          X_DATA(I)=TONE+(TTWO-TONE)*
     +    (FLOATJ(I-1)/FLOATJ(NUM_POINTS))

      ENDDO

      CALL READ(NUM_POINTS,X_DATA,Y_DATA,FILES(1))

      CALL FIRST_PLOT(DEVICE,NUM_POINTS,X_DATA,Y_DATA,
     +     X_TITLE,X_UNITS,Y_TITLE,Y_UNITS,TITLE,INFO)

      DO I=2,NUM_FILES


          CALL READ(NUM_POINTS,X_DATA,Y_DATA,FILES(I))
          CALL PLOT(DEVICE,NUM_POINTS,X_DATA,Y_DATA,INFO)

      ENDDO

      CALL PCLOSP
```

END

```
****************************************************************
*     Department of Electrical and Computer Engineering       *
*                 Kansas State University                      *
*                                                              *
*     VAX FORTRAN source filename:  PLOT_O_MATIC.FOR           *
****************************************************************
*
*   ROUTINE:              subroutine
*                         PLOT(DEVICE,NUM_POINTS,X_DATA,
*                         Y_DATA,INFO)
*
*
*     DESCRIPTION:        Makes a plot using X_DATA as abscissa
*                         and Y_DATA as ordinate.  The axes are
*                         assumed to be already drawn in
*                         accordance with INFO.
*
*
*     DOCUMENTATION
*     FILES:              None.
*
*
*     ARGUMENTS:
*
*         DEVICE          (input)  integer
*                         is the device type to display the plot
*
*                         7475     for plotter
*                         4014     for terminal (Selanar only)
*
*         NUM_POINTS      (input)  integer
*                         is the number of data points to
*                         be plotted
*
*         X_DATA          (input)  real
*                         is the array of abscissa values for the
*                         data to be plotted
*
*         Y_DATA          (input)  real
*                         is the array of ordinate values for the
*                         data to be plotted
*
*
*     RETURN:
*
*         INFO            (output)  real(6)
*                         is the information necessary to make
*                         subsequent plots on the same axes.
*
```

```
*
*      ROUTINES
*      CALLED:          P System of Generalized Plot Routines
*
*
*      AUTHOR:          James F. Stafford
*
*
*      DATE CREATED:    24May86   Version 1.0
*
*
*      REVISIONS:       None.
*
*
****************************************************************

      SUBROUTINE  PLOT(DEVICE,NUM_POINTS,X_DATA,Y_DATA,
     +                 INFO)

      IMPLICIT NONE

      INTEGER  DEVICE,NUM_POINTS,FORLAB,FORTIC,NEGFLG,FORM,
     +         SCNTL,LENSTR,UPDOWN

      REAL     X_DATA(*),Y_DATA(*),FACTOR,VEL,X,Y,LENGTH,
     +         FIRSTX,DELTAX,ANGLE,CLEN,FIRDEL(4),
     +         DIVLNX,DIVLNY,WIDTH,HEIGHT,INFO(6)

      CHARACTER*(1)  BLANK,SIZE

*INTIALIZE PLOT DEVICE

      FACTOR=1.0
      BLANK=' '
      SIZE='A'

*     CALL  PINIT(DEVICE,BLANK,FACTOR,SIZE)

*SET PEN VELOCITY

      VEL=10.0

      CALL  PSTVEL(VEL)

*ESTABLISH ORIGIN

      X=4.5
      Y=4.5
```

```
      CALL PORIG(X,Y)
*SET OFFSETS FOR AXIS ROUTINES (RELATIVE TO ORIGIN)
      X=0.0
      Y=0.0
*ESTABLISH INFORMATION FOR PLOTTING SUBROUTINE
      FIRDEL(1)=INFO(1)
      FIRDEL(2)=INFO(2)
      FIRDEL(3)=INFO(3)
      FIRDEL(4)=INFO(4)
      DIVLNX=INFO(5)
      DIVLNY=INFO(6)
*DRAW CURVE
          SCNTL=0
          CALL PLINE(X_DATA,Y_DATA,NUM_POINTS,FIRDEL,SCNTL,
     +    BLANK,DIVLNX,DIVLNY)
      RETURN
      END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:    POLADD.FOR               *
*****************************************************************
*
*      ROUTINE:           SUBROUTINE
*                         POLADD(A,DEGA,B,DEGB)
*
*
*      DESCRIPTION:       This program subtracts on polynomial
*                         from another, replacing the first addend
*                         with the difference.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:         The following arguments are passed to
*                         the subroutine.
*
*         A               (input/output) real
*                         is an array containing the coefficients
*                         for a polynomial on input and the
*                         coefficients for the difference on
*                         return.
*
*         DEGA            (input/output) integer
*                         is the degree of the above polynomial
*                         on input and the degree of the
*                         difference on return.
*
*         B               (input) real
*                         is an array containing the coefficients
*                         of the polynomial to be subtracted
*                         from A.
*
*         DEGB            (input) integer
*                         is the degree of the above polynomial.
*
*
*      RETURN:            Not used.
*
*
*      ROUTINES
*      CALLED:            None.
*
*
```

```
*       AUTHOR:         James F. Stafford
*
*
*       DATE CREATED:   6Jun87  Version 1.0
*
*
*       REVISIONS:      None.
*
*
****************************************************************
        SUBROUTINE      POLADD(A,DEGA,B,DEGB)

        IMPLICIT        NONE

        INTEGER         I,DEGA,DEGB,ADDS

        REAL*8          A(0:*),B(0:*)

        ADDS=0

        IF (DEGA.GE.DEGB) THEN

            DO I=0,DEGB

                A(I)=A(I)-B(I)
                ADDS=ADDS+1

            ENDDO

        ELSE

            DO I=0,DEGA

                A(I)=A(I)-B(I)
                ADDS=ADDS+1

            ENDDO

            DO I=DEGA+1,DEGB

                A(I)=-B(I)
                ADDS=ADDS+1

            ENDDO

        ENDIF

        DEGA=JMAX0(DEGA,DEGB)

        PRINT *,ADDS,'additions'
```

```
RETURN
END
```

```
*****************************************************************
*       Department of Electrical and Computer Engineering       *
*                  Kansas State University                      *
*                                                               *
*       VAX FORTRAN source filename:    POLDIV.FOR              *
*****************************************************************
*
*       ROUTINE:            SUBROUTINE
*                           POLDIV(NUM,N_DEG,DEN,D_DEG,QUO,DEGQ,
*                           REM,DEGR,EASY,HARD)
*
*
*       DESCRIPTION:        This program performs the division
*                           algorithm on two input polynomials
*                           to obtain a quotient and remainder.
*
*
*       DOCUMENTATION
*       FILES:              None.
*
*
*       ARGUMENTS:          The following arguments are passed to
*                           the subroutine.
*
*           NUM             (input) real
*                           is an array containing the coefficients
*                           of the numerator polynomial.
*
*           N_DEG           (input) integer
*                           is the degree of the numerator polynomial.
*
*           DEN             (input) real
*                           is an array containing the coefficients
*                           of the denominator polynomial.
*
*           D_DEG           (input) integer
*                           is the degree of the denominator
*                           polynomial.
*
*           QUO             (output) real
*                           is an array containing the coefficients
*                           of the quotient polynomial.
*
*           DEGQ            (output) integer
*                           is the degree of the quotient polynomial.
*
*           REM             (output) real
*                           is an array containing the coefficients
*                           of the remainder polynomial.
*
```

```
*           DEGR          (output) integer
*                         is the degree of the remainder
*                         polynomial.
*
*           EASY          (input) logical
*                         should be set to .TRUE. if both the
*                         numerator and denominator are monic
*                         polynomials.  This will save calculations.
*
*           HARD          (input) logical
*                         must be set to .TRUE. if the
*                         denominator is not a monic polynomial.
*                         Otherwise leave it false to save
*                         calculations.
*
*
*       RETURN:           Not used.
*
*
*       ROUTINES
*       CALLED:           None.
*
*
*       AUTHOR:           James F. Stafford
*
*
*       DATE CREATED:     6Jun87   Version 1.0
*
*
*       REVISIONS:        27Jul87 Added calculation-saving.
*
*
*****************************************************************
        SUBROUTINE        POLDIV(NUM,N_DEG,DEN,D_DEG,QUO,DEGQ,
     +                    REM,DEGR,EASY,HARD)

        IMPLICIT          NONE

        INTEGER           J,K,N_DEG,D_DEG,DEGQ,DEGR,MULT,ADD,DIV

        REAL*8            NUM(0:*),DEN(0:*),QUO(0:*),REM(0:*),ZERO

        LOGICAL           EASY,HARD

        PARAMETER         (ZERO=1.0E-5)

        DEGQ=N_DEG-D_DEG
        MULT=0
        DIV=0
        ADD=0
```

```
DO J=0,N_DEG

    REM(J)=NUM(J)

ENDDO

IF (DEGQ.GE.0) THEN

    DO K=DEGQ,0,-1

        IF (HARD) THEN

            QUO(K)=REM(D_DEG+K)/DEN(D_DEG)
            DIV=DIV+1

        ELSE

            QUO(K)=REM(D_DEG+K)

        ENDIF

        DO J=D_DEG+K-1,K,-1

            IF (EASY) THEN

                REM(J)=REM(J)-DEN(J-K)
                ADD=ADD+1

            ELSE

                REM(J)=REM(J)-QUO(K)*DEN(J-K)
                ADD=ADD+1
                MULT=MULT+1

            ENDIF

        ENDDO

        EASY=.FALSE.

    ENDDO

    IF (D_DEG.EQ.0) REM(0)=0.

    DEGR=JMAX0(0,D_DEG-1)

    DO WHILE ((DABS(REM(DEGR)).LT.ZERO).AND.DEGR.GT.0)

        DEGR=DEGR-1
```

```
          ENDDO

      ELSE

          DEGR=N_DEG
          DEGQ=0
          QUO(0)=0.

      ENDIF

*     PRINT *,MULT,'multiplies'
*     PRINT *,ADD,'additions'
*     PRINT *,DIV,'divisions'

      RETURN

      END
```

```
*******************************************************************
*      Department of Electrical and Computer Engineering         *
*                   Kansas State University                      *
*                                                                *
*      VAX FORTRAN source filename:    POLMULT.FOR               *
*******************************************************************
*
*      ROUTINE:            SUBROUTINE
*                          POLMULT(POL1,DEG1,POL2,DEG2,PROD,DEGP,
*                          EASY)
*
*
*      DESCRIPTION:        This program multiplies two polynomials
*                          and returns their product.
*
*
*      DOCUMENTATION
*      FILES:              None.
*
*
*      ARGUMENTS:          The following arguments are passed to
*                          the subroutine.
*
*          POL1,POL2       (input) real
*                          are arrays containing the coefficients
*                          of the polynomials to be multiplied.
*
*          DEG1,DEG2       (input) integer
*                          are the degrees of the above
*                          polynomials.
*
*          PROD            (output) real
*                          is an array containing the coefficients
*                          of the product polynomial.
*
*          DEGP            (output) integer
*                          is the degree of the product
*                          polynomial.
*
*          EASY            (input) logical
*                          should be set to .TRUE. only if POL1
*                          is monic in order to save calculations.
*
*
*      RETURN:             Not used.
*
*
*      ROUTINES
*      CALLED:             None.
*
```

```
*
*       AUTHOR:           James F. Stafford
*
*
*       DATE CREATED:     7Jun87  Version 1.0
*
*
*       REVISIONS:        20Jul87 Added calculation-saving.
*
*
*****************************************************************
        SUBROUTINE        POLMULT(POL1,DEG1,POL2,DEG2,PROD,DEGP,EASY)

        IMPLICIT          NONE

        INTEGER           J,K,DEG1,DEG2,DEGP,ADD,MULT

        REAL*8            POL1(0:*),POL2(0:*),PROD(0:*)

        LOGICAL           EASY

        ADD=0
        MULT=0
        DEGP=DEG1+DEG2

        DO J=DEG2,0,-1

            IF (EASY) THEN

                PROD(DEG1+J)=POL2(J)

            ELSE

                PROD(DEG1+J)=POL1(DEG1)*POL2(J)
                MULT=MULT+1

            ENDIF

        ENDDO

        DO J=DEG1-1,0,-1

            DO K=DEG2,1,-1

                PROD(J+K)=PROD(J+K)+POL1(J)*POL2(K)
                ADD=ADD+1
                MULT=MULT+1

            ENDDO
```

```
          PROD(J)=POL1(J)*POL2(0)
          MULT=MULT+1

      ENDDO

      PRINT *,ADD,'additions'
      PRINT *,MULT,'multiplies'

      RETURN

      END
```

```
*******************************************************************
*      Department of Electrical and Computer Engineering         *
*                  Kansas State University                       *
*                                                                *
*      VAX FORTRAN source filename:    POLY_READ.FOR             *
*******************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        POLY_READ(POLY,DEG)
*
*
*      DESCRIPTION:      This program reads a polynomial from
*                        a data file.  The user must enter
*                        the data file name from the keyboard.
*
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        The following arguments are passed to
*                        the subroutine.
*
*         POLY           (output) real
*                        is an array to receive the coefficients
*                        of the polynomial.
*
*         DEG            (output) integer
*                        is the degree of the polynomial.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           None.
*
*
*      AUTHOR:           James F. Stafford
*
*
*      DATE CREATED:     6Jun87  Version 1.0
*
*
*      REVISIONS:        None.
*
*
*******************************************************************
       SUBROUTINE        POLY_READ(POLY,DEG)
```

```
IMPLICIT        NONE
INTEGER         DEG,I
REAL*8          POLY(0:10)
READ (1,*) DEG
READ (1,*) (POLY(I), I=0,DEG)
RETURN
END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:    READ.FOR                 *
*****************************************************************
*
*      ROUTINE:          SUBROUTINE
*                        READ(NUM_POINTS,X_DATA,Y_DATA,FILENAME)
*
*
*      DESCRIPTION:      This program evaluates a response
*                        function computed by the inverse
*                        transform program.  The particular
*                        function parameters are stored in a
*                        data file under FILENAME.
*
*      DOCUMENTATION
*      FILES:            None.
*
*
*      ARGUMENTS:        The following arguments are passed to
*                        the subroutine:
*
*          NUM_POINTS    (input) integer
*                        is the number of points in X_DATA
*                        at which response values are desired.
*
*          X_DATA        (input) real
*                        is an array containing the
*                        values of time that the response
*                        function is to be evaluated at.
*
*          Y_DATA        (output) real
*                        is an array to accumulate the
*                        computed function values.
*
*          FILENAME      (input) character
*                        is the filename of the response
*                        function to be evaluated.
*
*
*      RETURN:           Not used.
*
*
*      ROUTINES
*      CALLED:           None.
*
*
*      AUTHOR:           James F. Stafford
```

```
*
*
*       DATE CREATED:    24May86 Version 1.0
*
*
*       REVISIONS:       None.
*
*
*****************************************************************
        SUBROUTINE       READ(NUM_POINTS,X_DATA,Y_DATA,FILENAME)

        IMPLICIT         NONE

        INTEGER          NUM_POINTS,I,J,K,NO_TERMS,MULTS

        REAL             X_DATA(1000),Y_DATA(1000),EXPONENTIAL,
     +                   TONE,TTWO,POWEROFT

        REAL*8           TAU,OMEGA,RESP(0:1)

        CHARACTER*(15)   FILENAME

        DO I=1,NUM_POINTS

            Y_DATA(I)=0.

        ENDDO

        OPEN (UNIT=1,FILE=FILENAME,STATUS='OLD')

        READ (1,*) NO_TERMS

        DO I=1,NO_TERMS

            READ(1,*) TAU,OMEGA
            READ(1,*) MULTS

            POWEROFT=1

            DO J=1,MULTS

                READ(1,*) RESP(0)
                READ(1,*) RESP(1)

                DO K=1,NUM_POINTS

                    IF (J-1.GT.0) POWEROFT=X_DATA(K)**(J-1)

                    Y_DATA(K)=Y_DATA(K)+
     +              SNGL(DEXP(-TAU*X_DATA(K))*
```

```
+                   (RESP(0)*DCOS(OMEGA*X_DATA(K))+
+                   RESP(1)*DSIN(OMEGA*X_DATA(K)))*
+                   POWEROFT

              ENDDO

         ENDDO

     ENDDO

     CLOSE (UNIT=1)

     RETURN

     END
```

```
*****************************************************************
*       Department of Electrical and Computer Engineering       *
*                   Kansas State University                     *
*                                                               *
*       VAX FORTRAN source filename:    ROOT_FIND.FOR           *
*****************************************************************
*
*       ROUTINE:          PROGRAM
*                         TEST
*
*       DESCRIPTION:      This program factors the denominator of
*                         a rational function with real coefficients
*                         into irreducible polynomials in R[x].
*                         The user is prompted for a filename
*                         under which a rational function has
*                         been stored.  The user is again prompted
*                         for a filename to store the result
*                         under.
*
*       DOCUMENTATION
*       FILES:            None.
*
*
*       ARGUMENTS:        None.
*
*
*       RETURN:           Not used.
*
*
*       ROUTINES
*       CALLED:           POLY_READ,FACTORER,SPEC_WRITE
*
*
*       AUTHOR:           James F. Stafford
*
*
*       DATE CREATED:     8Jun87   Version 1.0
*
*
*       REVISIONS:        None.
*
*
*****************************************************************
        PROGRAM           TEST

        IMPLICIT          NONE

        INTEGER           DEGN,DEGD,DEGF(10),NUM_FACTS,MULT(10),I,J

        REAL*8            NUM(0:10),DENOM(0:10),FACTOR(10,0:2)
```

```
      CHARACTER*15    FILENAME
      PRINT *,'Input data file name'
      READ (*,200) FILENAME
200   FORMAT (A15)
      OPEN (UNIT=1,FILE=FILENAME,STATUS='OLD')
      CALL POLY_READ(NUM,DEGN)
      CALL POLY_READ(DENOM,DEGD)
      CLOSE (UNIT=1)
      CALL FACTORER(DENOM,DEGD,FACTOR,NUM_FACTS,DEGF,MULT)
      DO I=1,NUM_FACTS
          PRINT *,'FACTOR NUMBER ',I
          DO J=0,DEGF(I)
              PRINT *,FACTOR(I,J)
          ENDDO
          PRINT *,'MULTIPLICITY',MULT(I)
      ENDDO
      CALL SPEC_WRITE(NUM,DEGN,NUM_FACTS,FACTOR,DEGF,MULT)
      END
```

```
*******************************************************************
*     Department of Electrical and Computer Engineering          *
*                Kansas State University                          *
*                                                                 *
*     VAX FORTRAN source filename:  SIMPLE_PLOT.FOR               *
*******************************************************************
*
*   ROUTINE:              subroutine
*                         SIMPLE_PLOT(DEVICE,NUM_POINTS,X_DATA,
*                         Y_DATA,X_AXIS_TITLE,X_AXIS_UNITS,
*                         Y_AXIS_TITLE,Y_AXIS_UNITS,PLOT_TITLE,
*                         PLOT_TYPE)
*
*
*      DESCRIPTION:       Makes a plot using X_DATA as abscissa
*                         and Y_DATA as ordinate.  The axes are
*                         labelled with titles and units.  The
*                         plot is also titled.  One of four plot
*                         types can be selected.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:
*
*          DEVICE         (input)  integer
*                         is the device type to display the plot
*
*                         7475     for plotter
*                         4014     for terminal (Selanar only)
*
*          NUM_POINTS     (input)  integer
*                         is the number of data points to
*                         be plotted
*
*          X_DATA         (input)  real
*                         is the array of abscissa values for the
*                         data to be plotted
*
*          Y_DATA         (input)  real
*                         is the array of ordinate values for the
*                         data to be plotted
*
*          X_AXIS_TITLE (input)  character*(*)
*                          is the title to be placed on the x_axis
*
*          X_AXIS_UNITS (input)  character*(*)
```

```
*                             is the name to be given to the units
*                             associated with the x-axis
*
*          Y_AXIS_TITLE (input)  character*(*)
*                             is the title to be placed on the y-axis
*
*          Y_AXIS_UNITS (input)  character*(*)
*                             is the name to be given to the units
*                             associated with the y-axis
*
*          PLOT_TITLE   (input)  character*(*)
*                             is the title to be placed on the plot
*
*          PLOT_TYPE    (input)  character*(*)
*                             is a character string which
*                             specifies the type of plot to be
*                             generated.  The following are valid:
*
*                             'LINEAR'      for linear-linear
*                             'LOG_LINEAR'  for log-linear
*                             'LINEAR_LOG'  for linear-log
*                             'LOG_LOG'     for log-log
*
*
*     RETURN:           Not used.
*
*
*     ROUTINES
*     CALLED:           P System of Generalized Plot Routines
*
*
*     AUTHOR:           James F. Stafford
*
*
*     DATE CREATED:     5Sep86   Version 1.0
*
*
*     REVISIONS:        None.
*
*
****************************************************************

      SUBROUTINE  SIMPLE_PLOT(DEVICE,NUM_POINTS,X_DATA,Y_DATA,
     +                   X_AXIS_TITLE,X_AXIS_UNITS,Y_AXIS_TITLE,
     +                  Y_AXIS_UNITS,PLOT_TITLE,PLOT_TYPE)

      IMPLICIT NONE

      INTEGER  DEVICE,NUM_POINTS,FORLAB,FORTIC,NEGFLG,FORM,
```

```
     +           SCNTL,LENSTR,UPDOWN

        REAL     X_DATA(*),Y_DATA(*),FACTOR,VEL,X,Y,LENGTH,
     +           FIRSTX,DELTAX,DIVLEN,ANGLE,CLEN,FIRDEL(4),
     +           DIVLNX,DIVLNY,WIDTH,HEIGHT

        CHARACTER*(*)  X_AXIS_TITLE,Y_AXIS_TITLE,X_AXIS_UNITS,
     +                 Y_AXIS_UNITS,PLOT_TITLE,PLOT_TYPE

        CHARACTER*(1)  BLANK,SIZE

*INTIALIZE PLOT DEVICE

        FACTOR=1.0
        BLANK=' '
        SIZE='A'

        CALL  PINIT(DEVICE,BLANK,FACTOR,SIZE)

*SET PEN VELOCITY

        VEL=10.0

        CALL  PSTVEL(VEL)

*ESTABLISH ORIGIN

        X=4.5
        Y=4.5

        CALL PORIG(X,Y)

*SET OFFSETS FOR AXIS ROUTINES (RELATIVE TO ORIGIN)

        X=0.0
        Y=0.0

*DRAW Y-AXIS AND LABEL

        IF (PLOT_TYPE.EQ.'LINEAR'.OR.PLOT_TYPE.EQ.'LINEAR_LOG')
     + THEN

           LENGTH=12.0

           CALL PSCALE(Y_DATA,NUM_POINTS,LENGTH,FIRSTX,
     +     DELTAX,DIVLEN)

           FIRDEL(3)=FIRSTX
           FIRDEL(4)=DELTAX
           DIVLNY=DIVLEN
```

```
          FORLAB=110
          FORTIC=1001
          ANGLE=90.0

          CALL  PAXIS(X,Y,Y_AXIS_TITLE,Y_AXIS_UNITS,FORLAB,
     +    FORTIC,LENGTH,ANGLE,FIRSTX,DELTAX,DIVLEN)

      ELSE

          LENGTH=12.0

          CALL PLOGSC(Y_DATA,NUM_POINTS,LENGTH,FIRSTX,CLEN,
     +    NEGFLG)

          FIRDEL(3)=FIRSTX
          FIRDEL(4)=CLEN
          FORM=-1010
          ANGLE=90.0

          CALL PLGAXS(X,Y,Y_AXIS_TITLE,Y_AXIS_UNITS,FORM,
     +    LENGTH,ANGLE,FIRSTX,CLEN)

      ENDIF

*DRAW X-AXIS AND LABEL

      IF (PLOT_TYPE.EQ.'LINEAR'.OR.PLOT_TYPE.EQ.'LOG_LINEAR')
     + THEN

          LENGTH=18

          CALL PSCALE(X_DATA,NUM_POINTS,LENGTH,FIRSTX,
     +    DELTAX,DIVLEN)

          FIRDEL(1)=FIRSTX
          FIRDEL(2)=DELTAX
          DIVLNX=DIVLEN
          FORLAB=211
          FORTIC=2001
          ANGLE=0.0

          CALL PAXIS(X,Y,X_AXIS_TITLE,X_AXIS_UNITS,FORLAB,
     +    FORTIC,LENGTH,ANGLE,FIRSTX,DELTAX,DIVLEN)

      ELSE
          LENGTH=18.0

          CALL PLOGSC(X_DATA,NUM_POINTS,LENGTH,FIRSTX,CLEN,
     +    NEGFLG)
```

```
          FIRDEL(1)=FIRSTX
          FIRDEL(2)=CLEN
          FORM=+2011
          ANGLE=0.0

          CALL PLGAXS(X,Y,X_AXIS_TITLE,X_AXIS_UNITS,FORM,
     +          LENGTH,ANGLE,FIRSTX,CLEN)

       ENDIF

*DRAW CURVE

       IF (PLOT_TYPE.EQ.'LINEAR')THEN

          SCNTL=0

          CALL PLINE(X_DATA,Y_DATA,NUM_POINTS,FIRDEL,SCNTL,
     +          BLANK,DIVLNX,DIVLNY)

       ELSE IF(PLOT_TYPE.EQ.'LOG_LINEAR')THEN

          SCNTL=0

          CALL PLGLIN(X_DATA,Y_DATA,NUM_POINTS,FIRDEL,SCNTL,
     +          BLANK,DIVLEN)

       ELSE IF(PLOT_TYPE.EQ.'LINEAR_LOG')THEN

          SCNTL=0

          CALL PLNLOG(X_DATA,Y_DATA,NUM_POINTS,FIRDEL,SCNTL,
     +          BLANK,DIVLEN)

       ELSE IF(PLOT_TYPE.EQ.'LOG_LOG')THEN

          SCNTL=0

          CALL PLGLOG(X_DATA,Y_DATA,NUM_POINTS,FIRDEL,SCNTL,
     +          BLANK)

       ENDIF

*TITLE THE PLOT

       UPDOWN=0
       X=9.0
       Y=13.0

       CALL PPLOT(X,Y,UPDOWN)
```

```
CALL PTXTLN(PLOT_TITLE,LENSTR)

WIDTH=-LENSTR/2
HEIGHT=0.0

CALL PCHRPL(WIDTH,HEIGHT)

CALL PTEXT(PLOT_TITLE)

CALL PCLOSP

RETURN
END
```

```
*********************************************************************
*       Department of Electrical and Computer Engineering          *
*                   Kansas State University                        *
*                                                                  *
*       VAX FORTRAN source filename:    SPEC_INPUT.FOR             *
*********************************************************************
*
*       ROUTINE:          PROGRAM
*                         SPEC_INPUT
*
*       DESCRIPTION:      This program allows the user to enter
*                         the specifications for a rational
*                         function already in factored form.
*
*       DOCUMENTATION
*       FILES:            None.
*
*
*       ARGUMENTS:        None.
*
*
*       RETURN:           Not used.
*
*
*       ROUTINES
*       CALLED:           None.
*
*
*       AUTHOR:           James F. Stafford
*
*
*       DATE CREATED:     15Jun87 Version 1.0
*
*
*       REVISIONS:        None.
*
*
*********************************************************************
        IMPLICIT          NONE

        INTEGER           I,J,N_DEG,D_DEG(10),NO_ROOTS,MULTS(10),
     +                    DEG,NO_FACTS

        REAL              NUM(0:15),DEN(0:2,10),X(0:1,10,5),
     +                    FACT(0:2),POLY(0:20)

        PRINT *,'Enter numerator information.'

        CALL INPUT_NONFACT(NUM,N_DEG)
```

```
PRINT *,'Enter denominator information.'
PRINT *,'Input number of relatively prime irreducible factors.'
READ (*,*) NO_FACTS
DO I=1,NO_FACTS
    PRINT *,'Enter information on factor number',I
    CALL INPUT_NONFACT(FACT,D_DEG(I))
    DO J=0,D_DEG(I)
        DEN(J,I)=FACT(J)
    ENDDO
    PRINT *,'Enter number of times this factor appears.'
    READ (*,*) MULTS(I)
ENDDO
CALL SPEC_WRITE(NUM,N_DEG,NO_FACTS,DEN,D_DEG,MULTS)
END
SUBROUTINE      INPUT_NONFACT(POLY,DEG)
IMPLICIT        NONE
INTEGER         DEG,I
REAL            POLY(0:*)
PRINT *,'Input degree'
READ (*,*) DEG
DO I=0,DEG
    PRINT *,'Input coeff. of power ',I
    READ (*,*) POLY(I)
ENDDO
RETURN
END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering        *
*                  Kansas State University                      *
*                                                               *
*      VAX FORTRAN source filename:    SPEC_READ.FOR            *
*****************************************************************
*
*      ROUTINE:           SUBROUTINE
*                         SPEC_READ(NUM,DEGN,DEN,DEGD,MULTS,
*                         NO_FACTS)
*
*
*      DESCRIPTION:       This program reads data for a rational
*                         function out of a file created by the
*                         program for factoring the denominator.
*                         There is also a program called
*                         SPEC_WRITE that will create a data file
*                         in the proper format.
*
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:         The following arguments are passed to
*                         the subroutine:
*
*          NUM            (output) real
*                         is an array containing the coefficients
*                         of the numerator polynomial.
*
*          DEGN           (output) integer
*                         is the degree of the numerator
*                         polynomial.
*
*          DEN            (output) real
*                         is a two-dimensional array.  DEN(I,J)
*                         represents the coefficient of the Ith
*                         power of x in the Jth factor of the
*                         denominator polynomial.
*
*          DEGD           (output) integer
*                         is an array.  DEGD(I) represents the
*                         degree of the Ith factor in the
*                         denominator polynomial.
*
*          MULTS          (output) integer
*                         is an array.  MULTS(I) represents the
*                         multiplicity of the Ith factor in the
```

```
*                        denominator polynomial.
*
*          NO_FACTS      (output) integer
*                        is the number of factors in the
*                        denominator polynomial.
*
*
*       RETURN:          Not used.
*
*
*       ROUTINES
*       CALLED:          None.
*
*
*       AUTHOR:          James F. Stafford
*
*
*       DATE CREATED:    6Jun87   Version 1.0
*
*
*       REVISIONS:       None.
*
*
**************************************************************
        SUBROUTINE       SPEC_READ(NUM,DEGN,DEN,DEGD,MULTS,NO_FACTS)

        IMPLICIT         NONE

        INTEGER          I,J,DEGN,DEGD(10),NO_FACTS,MULTS(10)

        REAL*8           NUM(0:15),DEN(0:2,10),X(0:1,10,5),
    +                    FACT(0:2)

        CHARACTER*15     FILENAME

        PRINT *,'Input data file name'

        READ (*,200) FILENAME

200     FORMAT (A15)

        OPEN (UNIT=1,FILE=FILENAME,STATUS='OLD')

        READ (1,*) DEGN

        DO I=0,DEGN

           READ (1,*) NUM(I)

        ENDDO
```

```
      READ (1,*) NO_FACTS
      DO I=1,NO_FACTS
          READ (1,*) DEGD(I)
          DO J=0,DEGD(I)
              READ (1,*) DEN(J,I)
          ENDDO
          READ (1,*) MULTS(I)
      ENDDO
      CLOSE (UNIT=1)
*     PRINT *,DEGN
*     DO I=0,DEGN
*         PRINT *,NUM(I)
*     ENDDO
*     PRINT *,NO_FACTS
*     DO I=1,NO_FACTS
*         PRINT *,DEGD(I)
*         DO J=0,DEGD(I)
*             PRINT *,DEN(J,I)
*         ENDDO
*         PRINT *,MULTS(I)
*     ENDDO
      RETURN
      END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering       *
*                   Kansas State University                    *
*                                                              *
*      VAX FORTRAN source filename:    SPEC_WRITE.FOR          *
*****************************************************************
*
*      ROUTINE:           SUBROUTINE
*                         SPEC_WRITE(NUM,N_DEG,NO_FACTS,FACTOR,
*                         D_DEG,MULTS)
*
*      DESCRIPTION:       This program writes out the factored
*                         form of a rational function to a file
*                         specified by the user.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:         The following arguments are passed to
*                         the subroutine:
*
*          NUM            (input) real
*                         is an array containing the coefficients
*                         of the numerator polynomial.
*
*          N_DEG          (input) integer
*                         is the degree of NUM
*
*          NO_FACTS       (input) integer
*                         is the number of factors in the
*                         denominator polynomial.
*
*          FACTOR         (input) real
*                         is an array containing the
*                         coefficients of each factor in
*                         the denominator polynomial.
*
*          D_DEG          (input) integer
*                         is an array specifying the
*                         degree of each factor in
*                         the denominator polynomial.
*
*          MULTS          (input) integer
*                         is an array specifying the
*                         multiplicity of each factor in
*                         the denominator polynomial.
*
*      RETURN:            Not used.
```

```
*
*
*       ROUTINES
*       CALLED:          None.
*
*
*       AUTHOR:          James F. Stafford
*
*
*       DATE CREATED:    30Jun88  Version 1.0
*
*
*       REVISIONS:       None.
*
*
*****************************************************************
        SUBROUTINE       SPEC_WRITE(NUM,N_DEG,NO_FACTS,FACTOR,D_DEG,MULTS

        IMPLICIT         NONE

        INTEGER          I,J,N_DEG,D_DEG(*),NO_FACTS,MULTS(*)

        REAL*8           NUM(0:*),FACTOR(10,0:2)

        CHARACTER*15     FILENAME

        PRINT *,'Enter filename.'

        READ (*,200) FILENAME

200     FORMAT (A15)

        OPEN (UNIT=1, FILE=FILENAME, STATUS='NEW')

        WRITE (1,*) N_DEG

        DO I=0,N_DEG

            WRITE (1,*) NUM(I)

        ENDDO

        WRITE (1,*) NO_FACTS

        DO I=1,NO_FACTS

            WRITE (1,*) D_DEG(I)

            DO J=0,D_DEG(I)
```

```
        WRITE (1,*) FACTOR(I,J)
    ENDDO
    WRITE (1,*) MULTS(I)
ENDDO
CLOSE (UNIT=1, STATUS='KEEP')
RETURN
END
```

```
*****************************************************************
*      Department of Electrical and Computer Engineering        *
*                   Kansas State University                     *
*                                                               *
*      VAX FORTRAN source filename:    TRANSFER.FOR             *
*****************************************************************
*
*      ROUTINE:           There are actually three programs
*                         in this file:
*                         SUBROUTINE
*                         GETD(J,A,DEGA,DEN,DEGD)
*
*                         SUBROUTINE
*                         GETX(J,K,F,DEGF,X,DEGX)
*
*                         SUBROUTINE
*                         PUTX(J,K,F,DEGF,X,DEGX)
*
*
*      DESCRIPTION:       These programs  are a substitute for
*                         a more sophisticated data structuring
*                         method.  They copy the coefficients
*                         for a polynomial embedded in a
*                         higher dimensional array into a one-
*                         dimensional array, or vice versa.
*
*
*      DOCUMENTATION
*      FILES:             None.
*
*
*      ARGUMENTS:         The following arguments are passed to
*                         GETD:
*
*         J               (input) integer
*                         is a number representing which factor
*                         of the denominator is sought.
*
*         DEN             (input) real
*                         is a two-dimensional array.  DEN(I,J)
*                         represents the coefficient of the Ith
*                         power of x in the Jth factor of the
*                         denominator polynomial.
*
*         DEGD            (input) integer
*                         is an array.  DEGD(I) represents the
*                         degree of the Ith factor in the
*                         denominator polynomial.
*
*         A               (output) real
```

```
*                     is an array to receive the coefficients
*                     of the factor polynomial.
*
*         DEGA        (output) integer
*                     is the degree of the factor polynomial.
*
*                     The following arguments are passed to
*                     GETX:
*
*         J,K         (input) integer
*                     are the coordinates of the term in the
*                     partial fraction expansion that is
*                     sought. See the description of X.
*
*         X           (input) real
*                     is a three-dimensional array.  X(I,J,K)
*                     represents the Ith coefficient of the
*                     numerator of the (J,K)th term in the
*                     partial fraction expansion.  Namely,
*                     that term with the Jth factor of DEN
*                     to the Kth power as denominator.
*
*         DEGX        (input) integer
*                     is an array.  DEGX(I,J) represents the
*                     degree of the numerator of the (I,J)th
*                     term in the partial fraction expansion.
*                     See the description of X.
*
*         F           (output) real
*                     is an array to receive the coefficients
*                     of the desired polynomial.
*
*         DEGF        (output) integer
*                     is the degree of the polynomial, F.
*
*                     The following arguments are passed to
*                     PUTX:
*
*         J,K         (input) integer
*                     are the coordinates of the term in the
*                     partial fraction expansion that is to
*                     be updated. See the description of X.
*
*         X           (input) real
*                     is a three-dimensional array.  X(I,J,K)
*                     represents the Kth coefficient of the
*                     numerator of the (I,J)th term in the
*                     partial fraction expansion.  Namely,
*                     that term with the Ith factor of DEN
*                     to the Jth power as denominator.
```

```
*
*         DEGX          (input) integer
*                       is an array.  DEGX(I,J) represents the
*                       degree of the numerator of the (I,J)th
*                       term in the partial fraction expansion.
*                       See the description of X.
*
*         F             (input) real
*                       is an array to containing the coefficients
*                       of the polynomial to be embedded in the
*                       partial fraction matrix.
*
*         DEGF          (input) integer
*                       is the degree of the polynomial, F.
*
*      RETURN:          Not used.
*
*
*      ROUTINES
*      CALLED:          None.
*
*
*      AUTHOR:          James F. Stafford
*
*
*      DATE CREATED:    6Jun87   Version 1.0
*
*
*      REVISIONS:       None.
*
*
*************************************************************
      SUBROUTINE       GETD(J,A,DEGA,DEN,DEGD)

      IMPLICIT         NONE

      INTEGER          J,K,DEGA,DEGD(*)

      REAL*8           A(0:*),DEN(0:2,*)

      DEGA=DEGD(J)

      DO K=0,DEGA

         A(K)=DEN(K,J)

      ENDDO

      RETURN
```

```
END
SUBROUTINE      GETX(J,K,F,DEGF,X,DEGX)
IMPLICIT        NONE
INTEGER         J,K,L,DEGF,DEGX(10,*)
REAL*8          F(0:*),X(0:1,10,*)
DEGF=DEGX(J,K)
DO L=0,DEGF
    F(L)=X(L,J,K)
ENDDO
RETURN
END
SUBROUTINE      PUTX(J,K,F,DEGF,X,DEGX)
IMPLICIT        NONE
INTEGER         J,K,L,DEGF,DEGX(10,*)
REAL*8          F(0:*),X(0:1,10,*)
DEGX(J,K)=DEGF
DO L=0,DEGF
    X(L,J,K)=F(L)
ENDDO
RETURN
END
SUBROUTINE      GET(A,DEGA,B,DEGB)
IMPLICIT        NONE
INTEGER         I,DEGA,DEGB
REAL*8          A(0:*),B(0:*)
```

```
DEGA=DEGB
DO I=0,DEGA
   A(I)=B(I)
ENDDO
RETURN
END
```

AN ALGEBRAIC APPROACH TO COMPUTING INVERSE LAPLACE TRANSFORMS OF RATIONAL FUNCTIONS

by

JAMES FLOYD STAFFORD

B.S. Kansas State University, 1986

AN ABSTRACT OF A MASTER'S THESIS
submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1989

Abstract

The usual method for solving linear, constant-coefficient. differential equations involves use of the Laplace transform. The most difficult step in this method of solution is computing the inverse Laplace transform of a rational function. The object of this thesis is to describe an algorithm for solving large systems of this kind. The thesis demonstrates that the problem of solving such systems can be treated completely algebraically once the denominator of the rational function is factored. It is shown that the number of operations required to compute a suitable partial fraction expansion of a rational function can be reduced by factoring the denominator into irreducible linear and quadratic factors in $\mathbf{R}[x]$. Applications to control theory are discussed. The algorithms are derived with mathematical rigor. Working FORTRAN programs implementing the derived algorithms are given in an appendix. Electrical engineers solving practical problems in circuit analysis and control theory might find these algorithms useful.