UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

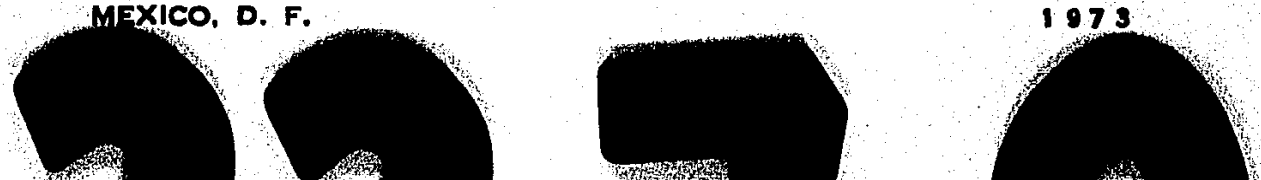
ALGUNOS ALGORITMOS PARA CALCULAR LAS RAICES DE UN POLINOMIO COMPLEJO

T E S I S

QUE PARA OBTENER
EL TITULO DE
ACTUARIO
PRESENTA

RAUL GUTIERREZ Y MONTERO

MEXICO, D. F. 1973



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

A MIS PADRES

I N D I C E

INTRODUCCION

Siendo miembro de un grupo asesor en Análisis Numérico que comenzaba a formarse dentro del CIMASS* bajo la dirección del Dr. Pablo Barrera Sánchez, empecé mi trabajo dentro del mismo estudiando algo acerca de la resolución de ecuaciones mediante procesos iterativos a principios del año pasado. Llegado el momento, vi muy someramente la aplicación del método de Newton en la resolución de ecuaciones polinomiales: no se hacía patente la diferencia entre la aplicación de dicho método al caso real y la aplicación al caso complejo.

Por aquel entonces el Dr. Barrera recibía el encargo del CIMASS de averiguar algo sobre un problema que se le presentaba al grupo de Probabilidad y Estadística de la misma institución: determinar las raíces de funciones analíticas. El Dr. Barrera notó la relación entre lo que yo estaba haciendo y dicho problema, pues existen criterios para la aproximación de funciones holomorfas mediante polinomios.

---

* Centro de Investigación en Matemáticas Aplicadas, Sistemas y Servicios.

A partir de aquel momento dediqué mi tiempo a documentarme sobre métodos que nos ayudaran en la resolución de polinomios en general con la intención de llegar a un buen método en la resolución de polinomios con coeficientes complejos. Manejamos varios métodos, más que nada para conocer sus ventajas y sus limitaciones. Así fue como nos enteramos de que los métodos de Newton–Raphson y Bairstow-Hitchcock poseían una convergencia magnífica pero sólo si las primeras aproximaciones eran lo suficientemente cercanas a la solución y que el método de Bernoulli nos resolvía en parte este problema al darnos una primera aproximación del cero de mayor (ó menor) magnitud de un polinomio.

Estudiamos entonces una generalización del método de Bernoulli: el método de Rutishauser ó Algoritmo Cociente–Diferencia (Q–D), que nos da una aproximación simultánea a todos los ceros de un polinomio y, más aun, de funciones trascendentes. Desgraciadamente este método resultaba, numéricamente, un poco inestable, es decir, se perdía mucha exactitud en los resultados debido a los errores de redondeo y tan sólo podíamos utilizarlo para obtener primeras aproximaciones a las raíces de un polinomio.

Pero fue uniendo todos los métodos que hemos enunciado hasta ahora como dimos un buen paso en la resolución de nuestro problema. Si los métodos de Newton y Bairstow funcionaban sólo con "buenas" primeras aproximaciones y el modelo Cociente–Diferencia era capaz de proporcionarnos esas primeras aproximaciones, lo lógico era unir en uno solo a estos tres métodos. Sin embargo, la mayor desventaja del método conjunto radicaba en su poca generalidad. Poca generalidad en el sentido de que la determinación de raíces que se presentaban en conjuntos de más de dos ceros del mismo módulo resultaba una labor muy complicada mediante este método, además de que sólo poseíamos su implementación para el caso de polinomios reales.

Casualmente, por aquel entonces, cayó en nuestras manos un artículo que no era más que un programa en FORTRAN para calcular las raíces de un

polinomio complejo (ref 11). Se basaba este programa en un artículo de M. A. Jenkins y J. F. Traub que describía un algoritmo –Iteración de tres pasos de desplazamiento variable– para calcular los ceros de un polinomio complejo (ref 10). Hicimos correr este programa para polinomios reales y complejos de reconocida dificultad en la determinación de sus raíces o especialmente construidos por nosotros, de grados y distribución de ceros arbitrarios. Todos los resolvió, con magníficas aproximaciones y tiempos de ejecución mínimos.

Profundizando en la teoría de este método nos dimos cuenta de su enorme poder, que superaba todas las desventajas mencionadas hasta ahora para los otros métodos. Esencialmente este es un método iterativo seguido de deflación, es decir, de los que Wilkinson considera muy confiables (ref 17).

El presente trabajo es la recopilación de lo hecho a lo largo de un año de estudio del tema que da título al mismo y creímos conveniente iniciarlo describiendo, en el capítulo 1, un método de localización que sirviera a manera de introducción, pues los métodos de ese tipo resultan sencillos e interesantes aunque, desde luego, no puedan competir en eficiencia con métodos como el de Rutishauser o el de Jenkins–Traub. Concretamente, pensamos en el método de Lehmer. Desgraciadamente, todos los escritos que al respecto consultamos (refs 2, 7, 12) se dedican a postular los resultados y posteriormente a checarlos. Aquí se da una versión original del algoritmo.

En el capítulo 2 pasamos al estudio del algoritmo Cociente–Diferencia, un método muy general en el sentido de sus numerosas aplicaciones, entre ellas la determinación de ceros y polos de funciones trascendentes, habiéndose derivado toda la teoría del método, incluyendo algunos aspectos enteramente nuevos, para este caso concreto y sólo hasta el final del capítulo dando el algoritmo que a nosotros nos interesa: es ahí donde se unen en un solo programa los métodos de Newton, Bairstow y Rutishauser. Finalmente, en el capítulo 3, hacemos un estudio completamente original del magnífico método de Jenkins y Traub.

Originalmente se pensó que mi tesis abarcara todo el problema mencionado en un principio, esto es, la determinación de raíces de funciones analíticas. Pero creo que en mi caso pasó lo que en muchos otros: a medida que uno avanza se va especializando más y más en determinadas partes del problema original. No obstante, pienso que se ha dado un buen paso en la resolución de éste. El siguiente sería la aproximación de funciones analíticas mediante funciones racionales, del cual se comienza a ver algo en el capítulo 2 de este trabajo, independientemente de que el método ahí discutido es bueno ya en la determinación de raíces de funciones trascendentes.

Por último, quiero agradecer al Dr. Pablo Barrera Sánchez la ayuda brindada no sólo durante el periodo de elaboración de esta tesis sino durante todo el tiempo que fui becario del CIMASS –tiempo del cual resultaron dos trabajos hechos conjuntamente con él– y mientras fui alumno suyo.

CAPITULO 1.

INTERPRETACION ALGORITMICA DEL TEOREMA DE ROUCHE PARA POLINOMIOS

1.0 INTRODUCCION

Comenzamos este trabajo discutiendo, en el presente capítulo, un algoritmo que a la vez que es interesante resulta de fácil comprensión. Nos referimos al método de Lehmer, un método de localización, de esos que tienen ciertos aspectos geométricos muy bonitos.

Hemos querido dar una versión original del algoritmo, partiendo de un hecho en el que creemos se funda todo el método —el teorema de Rouché— y llegando a establecer los resultados finales de una manera natural.

1.1 INDICE DE UN PUNTO CON RESPECTO A UNA CURVA CERRADA

1.1.1 Definición de índice. Consideremos una curva cerrada γ lisa por pedazos y asociémosle a cada punto a del plano complejo que no se encuentre sobre dicha curva un número entero de la siguiente manera: si el punto se encuentra en la región no acotada por la curva asociémosle el número cero, si se encuentra en una de las regiones acotadas y es "rodeado" por γ m veces en el sentido contrario al de las manecillas del reloj y n veces en el sentido inverso asociémosle el entero m-n. El número así asignado al punto a del plano complejo es llamado el índice del punto a con respecto a la curva cerrada γ y lo denotaremos por $n(\gamma, a)$.

Ejemplo. Los puntos localizados en la región no acotada Ω_1 de la figura 1.1 tienen por índice, con respecto a la curva cerrada γ, a 0, los de la región Ω_2 a 1 y aquéllos de la región Ω_3 a 2.

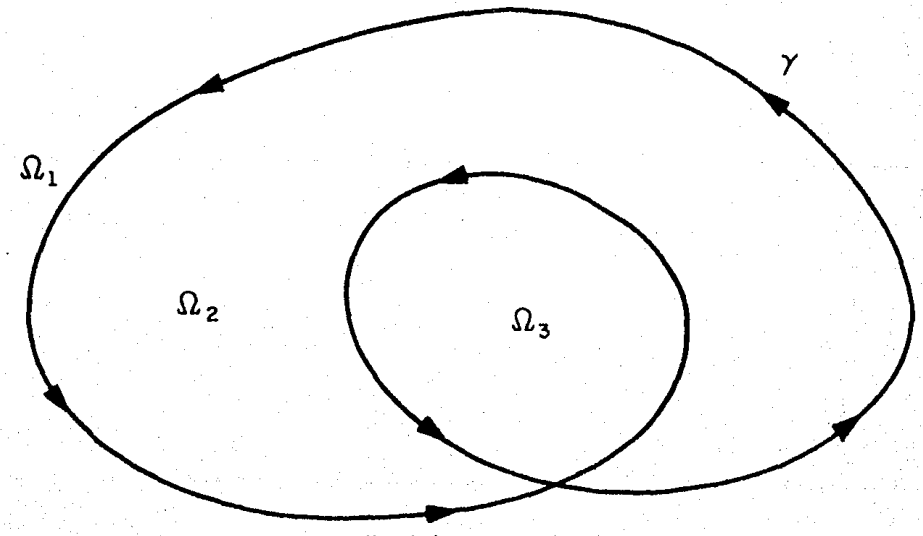


Fig 1.1

Si cambiásemos el sentido de la curva γ, los índices de dichos puntos nos vendrían dados por 0, −1, −2, respectivamente.

1.1.2 Expresión Analítica para el Indice de un Punto. Introduzcamos formalmente el concepto motivado en el párrafo anterior.

Lema 1.1. Si la curva cerrada γ lisa por pedazos no pasa por el punto a, entonces el valor de la integral

$$\int_{\gamma} \frac{dz}{z - a}$$

es múltiplo entero de $2\pi i$.

Demostración. Si la ecuación de γ es $z = z(t)$, $\alpha \leqslant t \leqslant \beta$, consideremos a la función

$$h(t) = \int_{\alpha}^{t} \frac{z'(t)}{z(t) - a}\, dt$$

Esta función está definida y es continua en el intervalo cerrado $[\alpha, \beta]$ y tiene por derivada a

$$h'(t) = \frac{z'(t)}{z(t) - a}$$

siempre que $z'(t)$ sea continua. De esta ecuación se sigue que la derivada de $e^{-h(t)}(z(t) - a)$ se anula excepto, quizá, en un número finito de puntos, y como esta función es continua debe reducirse a una constante. Así, tenemos que

$$e^{h(t)} = \frac{z(t) - a}{z(\alpha) - a}$$

Pero como $z(\beta) = z(\alpha)$, obtenemos $e^{h(\beta)} = 1$ y por lo tanto $h(\beta)$ debe ser un múltiplo de $2\pi i$. Esto prueba el lema.

Puede verse que $(1/2\pi i) \int_\gamma dz/(z - a)$ no es más que la cantidad que motivamos en el párrafo anterior, así que definiremos el índice del punto a con respecto a la curva cerrada γ por la ecuación

$$n(\gamma, a) = \frac{1}{2\pi i} \int_\gamma \frac{dz}{z - a}$$

Enunciamos ahora algunas propiedades de este índice (ref 1).

i) $n(-\gamma, a) = - n(\gamma, a)$ (como puede observarse del sencillo ejemplo dado en el párrafo anterior).

ii) Si γ se encuentra en el interior de un círculo, entonces $n(\gamma, a) = 0$ para todos los puntos que se encuentren en el exterior del mismo círculo.

iii) Como función de a el índice $n(\gamma, a)$ es constante en cada una de las regiones acotadas por γ y cero en la región no acotada.

1.2 CEROS DE UNA FUNCION ANALITICA EN EL INTERIOR DE UN CIRCULO Y EN UN SEMIPLANO

1.2.1 Interpretación Geométrica. Consideremos a la función analítica f(z) y supongamos que tiene un cero en el interior del círculo C.

La función $w = f(z)$ manda a la curva C en la curva cerrada Γ en el plano w; entonces, como f(z) tiene un cero en el interior de C, Γ rodea una vez al origen.

La interpretación geométrica ya no es tan obvia para cuando f(z)

tiene un par de raíces en el interior de C: sin embargo, no resulta muy complicado. Para ello, dividamos el círculo C en dos curvas simples cerradas, una conteniendo en su interior al cero z_1 de f(z) y la otra al cero z_2 tal y como se muestra en la figura 1.2. Pero por lo dicho para el caso anterior, cada una de estas curvas debería tener por imagen, bajo la función w = f(z), a una curva cerrada que rodeara una vez al origen, por lo que el caso que se ilustra en la misma figura 1.2 no es posible.

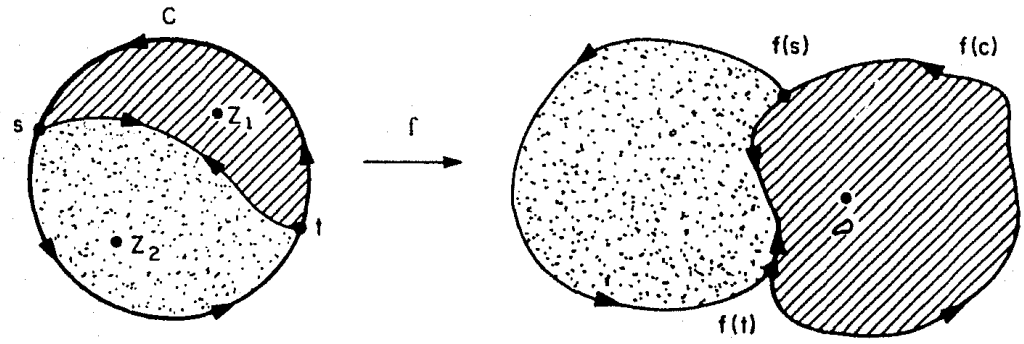


Fig 1.2

De la misma manera se ve que otros casos no son posibles, por lo que la única situación factible y natural debe ser del tipo de la que se ilustra en la figura 1.3. Es decir, la imagen de C bajo f seguramente rodea al origen dos veces.

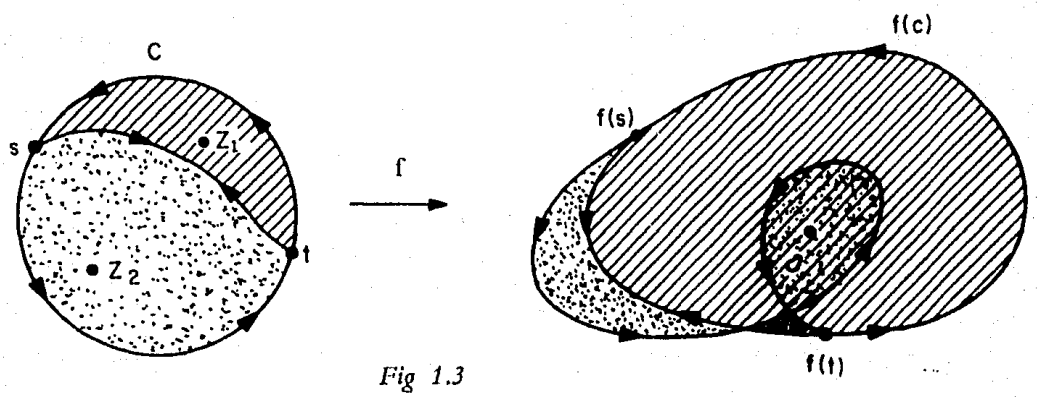


Fig 1.3

En general, si f(z) es una función analítica de raíces $z_1, z_2, \ldots, z_n$, de las cuales $z_1, z_2, \ldots, z_r$ se encuentran en el interior del círculo C, lo lógico es esperar que

la imagen Γ de C bajo f rodee p veces al origen en el sentido contrario al de las manecillas del reloj.

Probemos analíticamente este resultado.

1.2.2 Resultado Analítico. De lo dicho en el párrafo anterior se desprende el siguiente resultado.

Lema 1.2. Sea f(z) una función analítica en todo el plano. Supongamos que f(z) tiene un número finito de ceros y denotémoslos por $z_1, z_2, \ldots, z_n$, donde cada cero se repite tantas veces como su multiplicidad lo indique. Sea γ una curva cerrada lisa por pedazos tal que $f(z) \neq 0$ sobre γ y denotemos por Γ (en el plano w) a la curva imagen de γ bajo la función f(z). Entonces

$$n(\Gamma, o) = \sum_{j=1}^{n} n(\gamma, z_j).$$

Demostración. Como ya sabemos

$$n(\Gamma, o) = \frac{1}{2\pi i} \int_{\Gamma} \frac{dw}{w - o} = \frac{1}{2\pi i} \int_{\Gamma} \frac{dw}{w},$$

que por un sencillo cambio de variable se transforma en

$$n(\Gamma, o) = \frac{1}{2\pi i} \int_{\Gamma} \frac{f'(z)}{f(z)} \, dz ;$$

pero,

$$f(z) = (z - z_1)(z - z_2) \ldots (z - z_n)\, g(z).$$

con g(z) analítica y $\neq 0$ en todo el plano. Así pues,

$$\frac{f'(z)}{f(z)} = \frac{1}{z - z_1} + \frac{1}{z - z_2} + \ldots + \frac{1}{z - z_n} + \frac{g'(z)}{g(z)} .$$

donde, como $g(z) \neq 0$ en todo el plano, $w = g(z)$ manda al conjunto compacto formado por la curva γ y su interior en el conjunto compacto formado por la curva Γ_1 y su interior, el cual no contiene al origen; es decir, el índice del origen con respecto a la curva Γ_1 es cero, o lo que es lo mismo

$$\int_{\Gamma_1} \frac{dw}{w - 0} = \int_{\Gamma_1} \frac{dw}{w} = \int_{\gamma} \frac{g'(z)}{g(z)}\, dz = 0,$$

de tal manera que

$$n(\Gamma, 0) = \frac{1}{2\pi i} \int_{\gamma} \frac{f'(z)}{f(z)}\, dz$$

$$= \frac{1}{2\pi i} \int_{\gamma} \left(\sum_{j=1}^{n} \frac{1}{z - z_j} \right) dz + \frac{1}{2\pi i} \int_{\gamma} \frac{g'(z)}{g(z)}\, dz$$

$$= \frac{1}{2\pi i} \sum_{j=1}^{n} \int_{\gamma} \frac{dz}{z - z_j}$$

$$= \sum_{j=1}^{n} n(\gamma, z_j),$$

como se quería probar.

Corolario 1.3. Si γ, en el lema anterior, es una curva simple cerrada C (el cículo unitario, por ejemplo) que contiene en su interior p ceros, $z_1, z_2, \ldots, z_p$ de f(z), entonces

$$n(\Gamma, 0) = p .$$

Demostración

$$n(\Gamma, 0) = \sum_{j=1}^{n} n\,(C, z_j)$$

$$= \frac{1}{2\pi i} \sum_{j=1}^{n} \int_C \frac{dz}{z - z_j}$$

$$= \frac{1}{2\pi i} \sum_{j=1}^{p} \int_C \frac{dz}{z - z_j} + \frac{1}{2\pi i} \sum_{j=p+1}^{n} \int_C \frac{dz}{z - z_j}$$

$$= \frac{1}{2\pi i} \sum_{j=1}^{p} \int_C \frac{dz}{z - z_j} = \frac{1}{2\pi i}\, p \cdot 2\pi i = p\ .$$

El resultado establecido en esta sección vale, obviamente, para el caso particular de un polinomio P(z) de grado n con coeficientes complejos.

Observación. El número

$$n(\gamma, a) = \frac{1}{2\pi i} \int_\gamma \frac{dz}{z - a}$$

se puede definir también como la variación del argumento de z − a cuando el punto z recorre γ una vez en el sentido contrario al de las manecillas del reloj, tal y como se muestra en la figura de abajo.

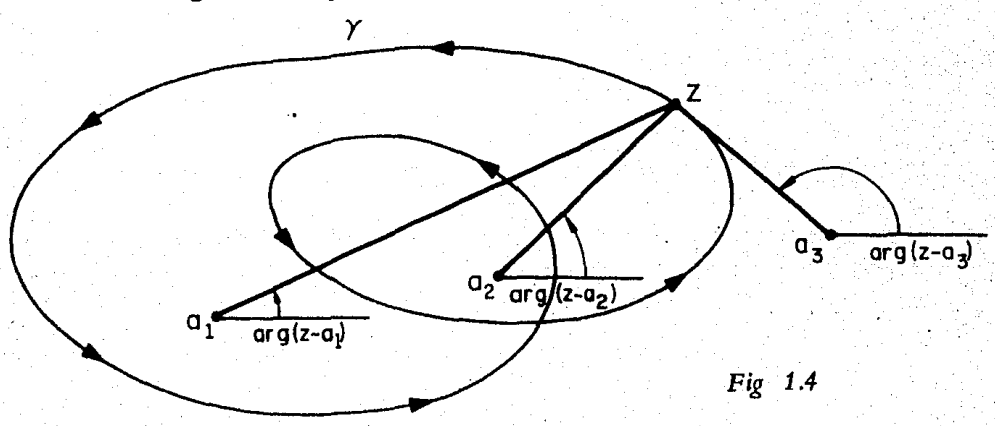


Fig 1.4

Como puede observarse de esta figura, la variación de los argumentos de $z - a_1$, $z - a_2$ y $z - a_3$ cuando z recorre γ una vez en el sentido contrario al de las manecillas del reloj es 2π, 4π y 0, respectivamente; es decir, que análogamente a como ocurre con el índice, si un punto a del plano complejo se encuentra en la región no acotada por la curva γ, la variación del argumento de $z - a$ cuando z recorre γ una vez en el sentido contrario al de las manecillas del reloj es cero; si, por el contrario, dicho punto se encuentra en una de las regiones acotadas y γ lo rodea m veces en el sentido contrario al de las manecillas del reloj y n veces en el sentido inverso, la variación del argumento de $z - a$ será 2π $(m - n)$.

Este hecho nos sirve para establecer el siguiente resultado.

Teorema 1.4. Sea L una recta en la que el polinomio complejo de grado n P(z) no tiene ceros. Denotemos por Δ_L arg P(z) la variación del argumento de P(z) cuando el punto z recorre L en una dirección específica y denotemos por p y q el número de ceros de P(z) a la izquierda y a la derecha de esta dirección de L, respectivamente. Entonces

$$p - q = \frac{1}{\pi} \Delta_L \arg P(z)$$

y así

$$p = \frac{1}{2} \cdot \left[n + \frac{1}{\pi} \Delta_L \arg P(z) \right],$$

$$q = \frac{1}{2} \left[n - \frac{1}{\pi} \Delta_L \arg P(z) \right].$$

Demostración. Si denotamos por $z_1, z_2, \ldots, z_p$ a los ceros de P(z) a la izquierda de la dirección en que el punto z recorre L y por $z_{p+1}, z_{p+1}, \ldots, z_n$ a aquéllos a la derecha de esta dirección y si

$$P(z) = a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n ,$$

entonces

$$P(z) = a_o (z - z_1)(z - z_2) \ldots (z - z_p)(z - z_{p+1}) \ldots (z - z_n),$$

donde

$$\arg P(z) = \arg a_o + \sum_{j=1}^{p} \arg (z - z_j) + \sum_{j=p+1}^{n} \arg (z - z_j).$$

Así, como puede verse de la interpretación geométrica que acabamos de dar, cuando el punto z recorre la recta L en el sentido indicado en la figura 1.5, el argumento de $z - z_j$ se incrementa en π cuando $1 \leqslant j \leqslant p$, pero disminuye π cuando $p < j \leqslant n$.

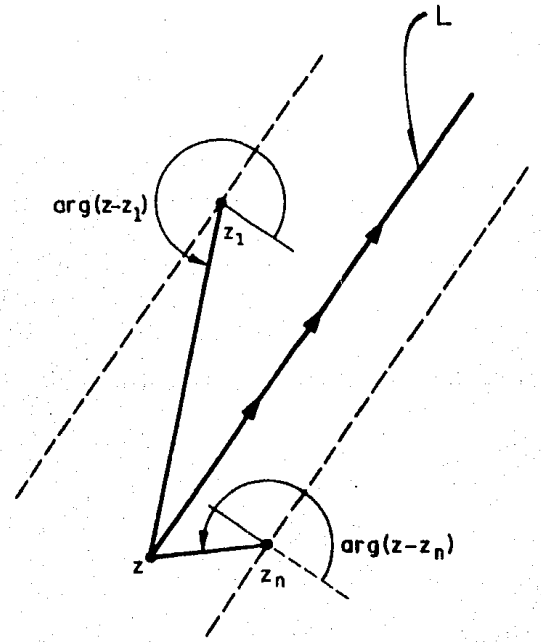


Fig 1.5

De esto se concluye que

$$\Delta_L \arg P(z) = p\pi - q\pi ,$$

$$p - q = \frac{1}{\pi} \Delta_L \arg P(z) ,$$

y como

$$p + q = n ,$$

entonces

$$p = \frac{1}{2}\left[n + \frac{1}{\pi}\Delta_L \arg P(z)\right] ,$$

$$q = \frac{1}{2}\left[n - \frac{1}{\pi}\Delta_L \arg P(z)\right] .$$

1.3 TEOREMA DE ROUCHE

Del primer resultado probado en la sección anterior se deriva el siguiente imporante teorema.

Teorema 1.5 (de Rouché). Si P(z) y Q(z) son funciones analíticas en el interior de una curva simple cerrada C y si son continuas en C y

$$|Q(z)| < |P(z)| \quad \text{en } C, \qquad (1.1)$$

entonces la función $F(z) = Q(z) + P(z)$ tiene el mismo número de ceros en el interior de C que P(z).

Demostración. Escribamos

$$F(z) = w\,P(z) , \quad w = 1 + \frac{Q(z)}{P(z)} . \qquad (1.2)$$

Si p denota el número de ceros de P(z) en el interior de C, entonces de acuerdo con el primer resultado de la sección anterior

$$p = \frac{1}{2\pi i} \int_C \frac{P'(z)}{P(z)}\, dz \,. \qquad (1.3)$$

Como $|Q(z)/P(z)| < 1$ en C, el punto w definido en las ecuaciones (1.2) describe (fig 1.6) una curva cerrada Γ que se encuentra en el interior del círculo con centro en $w = 1$ y radio 1. Así, el punto w permanece siempre en el semiplano derecho. Por lo tanto, el índice del origen con respecto a la curva Γ es cero. Esto significa

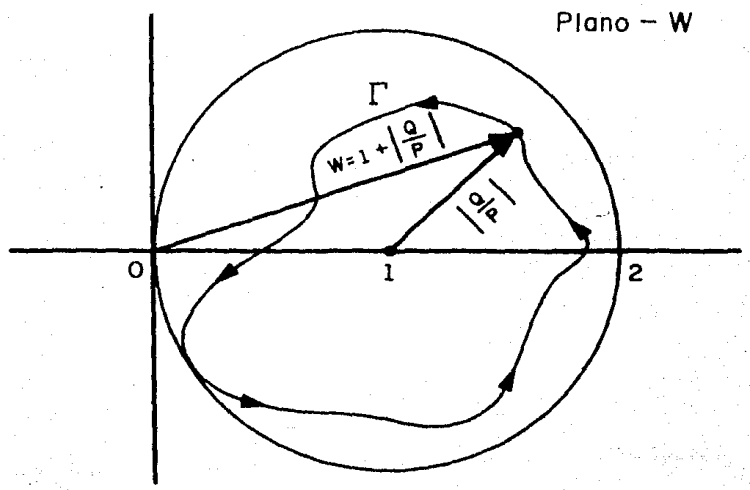


Fig 1.6

de acuerdo con las ecuaciones (1.2) y (1.3) que

$$\frac{1}{2\pi i} \int_C \frac{F'(z)}{F(z)}\, dz = \frac{1}{2\pi i} \int_\Gamma \frac{dw}{w} + \frac{1}{2\pi i} \int_C \frac{P'(z)}{P(z)}\, dz$$

$$= \frac{1}{2\pi i} \int_C \frac{P'(z)}{P(z)}\, dz = p$$

y de acuerdo con el resultado de la sección anterior F(z) tiene también p ceros en el interior de C.

1.4 INTERPRETACION ALGORITMICA DEL TEOREMA DE ROUCHE PARA POLINOMIOS

Estamos interesados en utilizar el teorema de Rouché para localizar y calcular las raíces del polinomio complejo

$$P(z) = a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n \, , \; a_o \neq 0 \, .$$

La idea básica es escoger un polinomio Q(z) del mismo grado que P(z) y tal que

i) Solamente se cumpla una de las siguientes desigualdades a lo largo de una curva cualquiera C,

$$|P(z)| < |Q(z)| \, ,$$

$$|P(z)| > |Q(z)| \, ;$$

ii) P(z) + Q(z) tiene cuando menos un cero conocido.

Veamos cómo podemos lograr lo anterior en dos casos particulares para C.

1.4.1 Ceros de un Polinomio Complejo en un Semiplano. Supongamos que la curva C del párrafo anterior es el eje real.

En este caso lo que necesitamos es que la función

$$h(z) = |P(z)| - |Q(z)|$$

sea positiva (ó negativa) para todo real z. Para hallar un polinomio Q(z) con tales

propiedades observemos lo siguiente: basta con hallar un polinomio q(z) de grado n tal que $|q(z)| = |P(z)|$, z real, para lograr nuestro primer objetivo, pues entonces tomaríamos $Q(z) = \beta\, q(z)$, con β un número complejo tal que $|\beta| < 1$ (o $|\beta| > 1$). El polinomio q(z) se puede obtener fácilmente a partir de P(z) de la siguiente manera

$$q(z) = \bar{a}_o z^n + \bar{a}_1 z^{n-1} + \ldots + \bar{a}_{n-1} z + \bar{a}_n = \overline{P(z)},$$

ya que así

$$q(z) = \overline{P(z)},$$

para z real.

Tendríamos entonces el siguiente lema.

Lema 1.6. Si P(z) y q(z) son polinomios complejos de grado n,

$$P(z) = a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n ,$$

$$q(z) = \bar{a}_o z^n + \bar{a}_1 z^{n-1} + \ldots + \bar{a}_{n-1} z + \bar{a}_n = \overline{P(z)},$$

y

$$F(z) = \alpha P(z) - \beta q(z) ,$$

con α y β números complejos tales que $|\alpha| > |\beta|$, entonces F(z) tiene el mismo número de ceros arriba del eje real que P(z) y el mismo número de ceros abajo del eje real que el mismo polinomio.

Supongamos, sin pérdida de generalidad, que P(z) no tiene ceros reales, pues si así fuera estos también serían ceros reales de q(z) por la forma en que

hemos definido a dicho polinomio y podríamos escribir

$$P(z) = P_1(z)\ r(z)\ ,$$

$$q(z) = q_1(z)\ r(z)\ ,$$

donde $r(z)$ es el polinomio cuyas únicas raíces son los ceros reales de $P(z)$, y trabajar entonces con

$$F_1(z) = \frac{F(z)}{r(z)} = \alpha P_1(z) - \beta q_1(z)\ .$$

Bajo esta suposición, todos los ceros de $F(z)$ son complejos pues si $F(z) = 0$, entonces

$$\alpha P(z) = \beta q(z) = \beta \overline{P}(z),$$

pero si z es real, $|P(z)| = |\overline{P}(z)|$, y de aquí $|\alpha| = |\beta|$, contrario a las hipótesis.

Ahora, como $|\beta| \geq 0$, $|\alpha| \neq 0$ y podemos considerar a

$$F(z) = \alpha \left[P(z) - \frac{\beta}{\alpha} q(z) \right], \quad \left| \frac{\beta}{\alpha} \right| < 1,$$

y la demostración del lema se sigue del teorema de Rouché para el caso en que la curva C es una recta.

El lema establece que si $|\alpha| > |\beta|$ en $F(z) = \alpha P(z) - \beta q(z)$, entonces este polinomio tiene el mismo número de ceros arriba y abajo del eje real que $P(z)$; esto nos sugiere escoger α y β de tal manera que $F(z)$ tenga un cero conocido, consiguiendo así el segundo de nuestros objetivos mencionados al principio de esta sección. La manera de lograr esto se muestra a continuación.

Trataremos de encontrar números complejos α y β de tal manera que el número complejo ξ, $\mathrm{Im}(\xi) < 0$, $|P(\xi)| \neq |\overline{P}(\xi)|$, sea raíz del polinomio

$$F(z) = \alpha(\xi)\, P(z) - \beta(\xi)\, \overline{P}(z),$$

es decir, tales que, para cuando $z = \xi$,

$$F(\xi) = \alpha(\xi)\, P(\xi) - \beta(\xi)\, \overline{P}(\xi) = 0.$$

Una solución nos vendría dada por

$$\alpha(\xi) = \overline{P}(\xi),\ \beta(\xi) = P(\xi),$$

presentándosenos dos casos:

i) $|\overline{P}(\xi)| > |P(\xi)|$,

ii) $|\overline{P}(\xi)| < |P(\xi)|$.

En el primer caso F(z) tendría, por el lema 1.6, el mismo número de ceros que P(z) arriba del eje x y, como $\mathrm{Im}(\xi) < 0$,

$$\frac{F(z)}{z - \xi} = P_1(z)$$

también, pero ya con $P_1(z)$ un polinomio de grado $n - 1$.

En el segundo caso, como

$$|\overline{P}(\xi)| < |P(\xi)|,$$

no nos conviene determinar a las constantes α y β de tal manera que ξ sea una raíz de F(z), sino de tal forma que en $z = \bar{\xi}$ F(z) tenga un cero, pues entonces tendríamos que para que

$$F(\bar{\xi}) = \alpha(\bar{\xi})\, P(\bar{\xi}) - \beta(\bar{\xi})\, \bar{P}(\bar{\xi}) = 0 ,$$

habría que escoger, por ejemplo,

$$\alpha(\bar{\xi}) = \bar{P}(\bar{\xi}) = \overline{P(\xi)} ,$$

$$\beta(\bar{\xi}) = P(\bar{\xi}) = \overline{\bar{P}(\xi)} ,$$

de donde

$$|\alpha(\bar{\xi})| > |\beta(\bar{\xi})| ,$$

de tal suerte que F(z) tendría el mismo número de ceros que P(z) arriba del eje x, y como ahora $\mathrm{Im}(\bar{\xi}) > 0$,

$$\frac{F(z)}{z - \bar{\xi}} = P_1(z) ,$$

tendría un cero menos que P(z) arriba del eje real.

Resumiendo, lo que hemos hecho en la presente sección es lo siguiente: Tomamos $P_0(z) = P(z)$ y formamos la "sucesión" de polinomios

$$P_0(z) , \quad P_1(z)$$

como sigue: Seleccionamos , de ser posible, un número complejo ξ tal que $|\bar{P}_0(\xi)| > |P_0(\xi)|$. Entonces tomamos

$$P_1(z) = \frac{\overline{P}_o(\xi)\, P_o(z) - P_o(\xi)\, \overline{P}_o(z)}{z - \xi}$$

Si $\mathrm{Im}(\xi) < 0$, entonces $P_1(z)$ tiene el mismo número de ceros arriba del eje real que $P_o(z)$. Si $\mathrm{Im}(\xi) > 0$, tiene uno menos. Más aún, para cualquier ξ dada para la cual $|P_o(\xi)| \neq |\overline{P}_o(\xi)|$, si $|P_o(\xi)| > |\overline{P}_o(\xi)|$, entonces $|P_o(\overline{\xi})| < |\overline{P}_o(\overline{\xi})|$.

El mismo procedimiento podría seguirse para $P_i(z)$ y polinomios subsecuentes y construir así una sucesión que nos permitiría elaborar un algoritmo para la localización de los ceros de P(z). Sin embargo, un algoritmo más simple se obtiene para el caso que a continuación analizamos.

1.4.2 Ceros de un Polinomio Complejo en el Interior del Círculo Unitario. Supongamos ahora que la curva C mencionada al principio de esta sección es el círculo unitario.

En este caso necesitamos encontrar un polinomio q(z) de grado n tal que

$$|q(z)| = |P(z)|.$$

para $|z| = 1$. Observemos que si $|z| = 1$, entonces

$$\begin{aligned} |P(z)| &= |a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n| \\ &= |z^n (a_o + a_1 z^{-1} + \ldots + a_{n-1} z^{-n+1} + a_n z^{-n})| \\ &= |z^n|\,|a_o + a_1 z^{-1} + \ldots + a_{n-1} z^{-n+1} + a_n z^{-n}| \\ &= |a_o + a_1 z^{-1} + \ldots + a_{n-1} z^{-n+1} + a_n z^{-n}|, \end{aligned}$$

y como

$$z = \bar{z}^{-1} ,$$

$$z^k = \bar{z}^{-k} ,$$

k natural, tendremos que

$$\begin{aligned} |P(z)| &= |a_o + a_1\bar{z} + \ldots + a_{n-1}\bar{z}^{n-1} + a_n\bar{z}^n| \\ &= \overline{|\bar{a}_o + \bar{a}_1 z + \ldots + \bar{a}_{n-1}z^{n-1} + \bar{a}_n z^n|} \\ &= |\bar{a}_o + \bar{a}_1 z + \ldots + \bar{a}_{n-1}z^{n-1} + \bar{a}_n z^n| \end{aligned}$$

de donde una posible elección para q(z) sería

$$q(z) = \bar{a}_n z^n + \bar{a}_{n-1}z^{n-1} + \ldots + \bar{a}_1 z + \bar{a}_o = z^n \bar{P}(z^{-1}) .$$

Entonces si P(z) y q(z) son polinomios complejos de grado n,

$$P(z) = a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n ,$$

$$q(z) = \bar{a}_n z^n + \bar{a}_{n-1}z^{n-1} + \ldots + \bar{a}_1 z + \bar{a}_o = z^n \bar{P}(z^{-1}) ,$$

y

$$F(z) = \alpha P(z) - \beta q(z) ,$$

con α y β números complejos tales que $|\alpha| > |\beta|$, tendremos, por el teorema de Rouché, que F(z) tiene el mismo número de ceros en el interior del círculo unitario que P(z).

Procediendo de manera análoga al caso anterior, se llega al requerido algoritmo. Esto lo hacemos en el siguiente párrafo.

1.4.3 El Algoritmo. Para terminar con este primer capítulo, describamos brevemente el algoritmo a que da lugar el teorema de Rouché. Para ello, en vez de asignarle un cero de antemano a F(z), procedimiento que se siguió en el primer caso, disminuiremos su grado. Veamos cómo es esto.

Consideremos al polinomio con coeficientes complejos

$$P(z) = a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n,\ a_o \neq 0,$$

y supongamos que $P(0) \neq 0$, pues en caso contrario podemos deflacionar inmediatamente el polinomio hasta obtener uno que no tenga a 0 por raíz. La combinación lineal

$$T(P) = \bar{a}_n P(z) - a_o P^*(z).$$

donde

$$P^*(z) = \bar{a}_n z^n + \bar{a}_{n-1} z^{n-1} + \ldots + \bar{a}_1 z + \bar{a}_o = z^n \bar{P}(z^{-1}),$$

no tiene término en z^n de tal manera que T(P) es de grado menor que P. El término constante de T(P) es de particular interés debido a que es real. En efecto

$$T(P(0)) = \bar{a}_n a_n - a_o \bar{a}_o = |a_n|^2 - |a_o|^2$$

Si este término constante es diferente de cero, podremos aplicar la transformación T al polinomio T(P). Procediendo de manera análoga con los polinomios subsecuentes obtenemos la sucesión finita

$$T(P),\ T^2(P),\ \ldots,\ T^k(P), \qquad (1.4)$$

donde

$$T^k (P(0)) = 0.$$

Si denotamos por d_i al grado de $T^i(P)$, tendremos que

$$n = d_o > d_1 > d_2 > \ldots > d_k \geq 0 .$$

El interés de la sucesión (1.4) radica en el siguiente teorema, que ha quedado establecido a todo lo largo del capítulo.

Teorema 1.7. Supóngase que $P(0) \neq 0$. Si, para alguna $h > 0$, $T^h(P(o)) < 0$, entonces P tiene al menos una raíz en el interior del círculo unitario C. Si, por el contrario, $T^i (P(0)) > 0$ para $1 \leq i < k$ y $T^{k-1}(P)$ es constante, entonces P no tiene raíces en el interior de C.

Este algoritmo básicamente se utiliza para encontrar aproximaciones numéricas a las ríces de un polinomio con coeficientes complejos y consiste en plantearse una pregunta básica: ¿tiene un polinomio dado una raíz en el interior de un círculo dado? Por ejemplo, supongamos que se tiene el círculo unitario y que el valor de nuestro polinomio en el origen es distinto de cero, i,e., $P(0) \neq 0$; se va duplicando entonces (o disminuyendo a la mitad) el radio de dicho círculo progresivamente hasta que se llega a uno de radio R tal que el anillo.

$$R < |z| < 2R$$

contiene una raíz de P(z) en su interior, mientras que el círculo interior se encuentra libre de raíces. Este anillo puede ser cubierto por ocho círculos traslapados de radio (5/6)R cada uno, uno de los cuales contendrá a una raíz de P(z). Esto cubre un primer paso. Enseguida procedemos de manera análoga con este último círculo (con centro en a_1) hasta llegar a un círculo de radio R_1 tal que el anillo

$$R_1 < |z - a_1| < 2R_1.$$

$$R_1 = \frac{5}{6} R\, 2^{-\theta}, \quad \theta \text{ natural},$$

contenga a una raíz de P(z) en su interior, pero de tal manera que el círculo interior no contenga raíces. De nuevo cubrimos este anillo con ocho círculos translapados, ahora de radio

$$\frac{5}{6} R_1 = \frac{5}{6} \left(\frac{5}{6} R\right) 2^{-\theta} = \frac{25}{36} R\, 2^{-\theta} \leq \frac{25}{72} R = 2 \left(\frac{5}{12}\right)^2 R$$

cada uno, uno de los cuales contendrá a una raíz de P(z). Esto completa el paso 2.

Es fácil ver que después de K pasos tendremos un círculo de radio menor o igual que $2\,(5/12)^K R$ que contenga una raíz de P(z). Es decir, este método no posee la convergencia cuadrática del método de Newton, pero sí la de la progresión geométrica de razón 5/12. Además, como se ve, las raíces son localizadas por orden creciente de magnitud.

Todo lo anterior vale para el problema: ¿tiene el polinomio Q(z) una raíz en el interior del círculo $|z - c| = \rho$?, pues este puede reducirse al caso del círculo unitario para el polinomio

$$P(z) = Q(\rho z + c),$$

ya que éste tiene una raíz $\beta = (a - c)/\rho$ para toda raíz a de Q y $|\beta| < 1$ si y sólo si $|a - c| < \rho$, i.e., si y sólo si a está en el interior del círculo $|z - c| = \rho$.

Para responder entonces a la pregunta ¿tiene un polinomio dado P(z) una raíz en el interior del círculo unitario C?, construímos la sucesión de polinomios (1.4) tal y como se indicó al principio de esta sección.

Para aclarar ideas terminemos con un sencillo ejemplo.

Ejemplo. Sea

$$P(z) = 9 + 3z - 14z^2 - 8z^3$$

Entonces

$$P^*(z) = -8 - 14z + 3z^2 + 9z^3.$$

de donde

$$\begin{aligned} T(P(z)) &= 9P(z) - (-8)P^*(z) \\ &= 17(1 - 5z - 6z^2) = 17P_1(z), \end{aligned}$$

con

$$P_1(z) = 1 - 5z - 6z^2.$$

De aquí $T(P(0)) > 0$. Ahora

$$P_1^*(z) = -6 - 5z + z^2,$$

por lo tanto

$$\begin{aligned} T^2(P(0)) &= 17T(P_1(0)) \\ &= 17[(1)P_1(0) - (-6)P_1^*(0)] \\ &= 17(-35) < 0. \end{aligned}$$

Así, P(z) tiene una raíz en el interior de C. En efecto, las raíces de este polinomio son 3/4, −1, −3/2, no habiendo importado que una de ellas está sobre C para poder aplicar el teorema 1.7.

Si en el polinomio original reemplazamos z por z/2, tendremos que

$$P(z) = 18 + 3z - 7z^2 - 2z^3,$$

para el que es fácil comprobar que

$$T(P(z)) = 320 + 40z - 120z^2 = 10P_1(z),$$

$$T^2(P(z)) = 10T(P_1(z)) = 8800 + 1760z = 440P_2(z),$$

$$T^3(P(z)) = 440T(P_2(z)) = 440(384),$$

donde

$$P_1(z) = 32 + 4z - 12z^2,$$

$$P_2(z) = 20 + 4z,$$

y de aquí

$$T(P(0)) = 320 > 0,$$

$$T^2(P(0)) = 8800 > 0,$$

$$T^3(P(0)) = 168960 > 0,$$

respectivamente, es decir $T^i(P(0)) > 0$, $i = 1, 2, 3$ y $T^3(P(z))$ es constante, de tal manera que por nuestro teorema, P(z) no tiene raíces en el interior de C. Con esto queremos hacer notar tan sólo que el polinomio original $P(z) = 9 + 3z - 14z^2 - 8z^3$ tiene cuando menos una raíz en el anillo

$$\frac{1}{2} < |z| < 1,$$

pero ninguna raíz en $|z| < \frac{1}{2}$.

La condición del teorema 1.7 de que T^{k-1} (P(z)) sea constante y el hecho de que nada se diga acerca de cuando T^{k-1} (P(z)) no sea constante parece ser una restricción grave del mencionado teorema. Sin embargo, no es así, pues el evento de que tal transformación no sea constante tiene probabilidad cero de ocurrir si los coeficientes de P(z) son números complejos aleatorios.

La implementación del algoritmo discutido a lo largo de este capítulo es posible, pero muy complicada, según se desprende de lo dicho por Lehmer en (ref 12): afirma que uno de los programas más simples que se logró hacer correr consta de 900 instrucciones. No obstante, en ese mismo artículo se da un diagrama de flujo conciso y claro.

Para nosotros, el interés principal del algoritmo aquí estudiado fue más que nada teórico, pues en los dos capítulos restantes veremos métodos más computables, eficientes y modernos y sus correspondientes implementaciones.

CAPITULO 2.

EL ALGORITMO COCIENTE–DIFERENCIA (Q–D)

2.0 INTRODUCCION

En el presente capítulo se estudiará un método debido a Rutishauser y que en realidad tiene muchas más aplicaciones de la que se intentará dar en la última sección (ref 4). Es por ello que la motivación, deducción y demostración del método se hace en general para la función meromorfa F(z) y la determinación de sus ceros y de sus polos, y sólo hasta la última sección es cuando se describe el algoritmo y se mide su eficiencia en el caso particular de la determinación de las raíces de un plinomio.

En la siguiente sección se prueba el teorema de Koenig y se motiva el algoritmo; en la segunda sección se aproxima la función F(z) mediante funciones racionales y con ellas se construye la tabla de Padé, viéndose la estrecha relación que ésta guarda con la afirmación del teorema de Koenig; en la parte final de la misma

sección se ve una segunda manera de obtener el algoritmo a partir de la tabla de Padé.

Finalmente, en la última sección, se describe un algoritmo útil para calcular las raíces de un polinomio complejo, sin embargo, como la implementación de tal algoritmo no es sencilla, la sección se concluye analizando la eficiencia de un programa que calcula las raíces de polinomios reales, incluyéndose el listado de dicho programa. Este se probó exhaustivamente y demostró ser bueno para polinomios de grados no muy grandes; aquí se incluyen sus resultados.

Como señalamos, el método es debido a Rutishauser, y otros autores lo han investigado a fondo (refs 4, 5, 7, 8 y 14). No obstante, el presente capítulo contiene resultados enteramente nuevos, en especial la parte intermedia del mismo.

2.1 TEOREMA DE KOENIG Y MOTIVACION DEL ALGORITMO COCIENTE–DIFERENCIA (Q–D)

2.1.1 Teorema de Koenig. Consideremos a la función meromorfa F(z) y supongamos que es halomorfa en z = 0. Se nos presenta entonces el siguiente problema: determinar las singularidades de dicha función a partir de los coeficientes de su desarrollo en serie de Taylor

$$F(z) = \sum_{j=0}^{\infty} c_j z^j , \quad c_0 \neq 0 \qquad (2.1)$$

alrededor del origen.

Veamos primeramente un par de casos particulares para entender la afirmación que queremos hacer y su correspondiente demostración.

Supongamos que F(z) tiene un polo r simple en $|z| < R$. Entonces

$$\Psi(z) = (z - r) F(z) = b_0 + b_1 z + b_2 z^2 + \ldots$$

es holomorfa en $|z| < R$ y

$$\begin{aligned} -\, r\, c_0 &= b_0 , \\ c_0 - r\, c_1 &= b_1 , \\ &\vdots \\ c_{n-1} - r\, c_n &= b_n , \\ &\vdots \end{aligned}$$

si multiplicamos las primeras $n+1$ ecuaciones por $1, r, r^2, \ldots, r^n$, respectivamente y las sumamos miembro a miembro a todas ellas, obtendremos

$$\Psi_n (r) = - c_n r^{n+1} , \qquad (2.2)$$

donde

$$\Psi_n (z) = b_0 + b_1 z + \ldots + b_n z^n . \qquad (2.3)$$

De aquí

$$\Psi_n (r) \underset{n \to \infty}{\longrightarrow} \Psi (r) , \qquad (2.4)$$

de donde

$$\frac{\Psi_n (r)}{\Psi_{n+1} (r)} \underset{n \to \infty}{\longrightarrow} 1 .$$

Pero $\Psi_n(r)/\Psi_{n+1} (r) = c_n/c_{n+1}\, r$, o equivalentemente

$$\begin{vmatrix} c_n & r\, \Psi_n (r) \\ c_{n+1} & \Psi_{n+1}(r) \end{vmatrix} = 0, \qquad (2.5)$$

por lo tanto, para n suficientemente grande,

$$\begin{vmatrix} c_n & r \\ c_{n+1} & 1 \end{vmatrix} \approx 0 , \qquad (2.6)$$

es decir,

$$\frac{c_n}{c_{n+1}} \xrightarrow[n \to \infty]{} r ,$$

que es precisamente lo que establece el teorema de Koenig en este primer caso.

Como segundo caso, supongamos que F(z) tiene dos polos simples r_1 , r_2 de módulos distintos y denotemos por r a cualquiera de ellos.

Sea

$$\varphi(z) = a_2 z^2 + a_1 z + 1 = (1 - r_1^{-1} z)(1 - r_2^{-1} z)$$

el polinomio que tiene por raíces a los polos de F. Entonces

$$\Psi(z) = \varphi(z) F(z) = b_0 + b_1 z + b_2 z^2 + \ldots$$

es holomorfa en $|z| < R$ y

$$\begin{aligned} c_0 &= b_0 , \\ c_1 + a_1 c_o &= b_1 , \\ c_2 + a_1 c_1 + a_1 c_0 &= b_2 , \\ &\vdots \\ c_n + a_1 c_{n-1} + a_2 c_{n-2} &= b_n , \\ &\vdots \end{aligned}$$

si multiplicamos de nuevo las primeras n + 1 ecuaciones por 1, r, r^2, . . . , r^n, respectivamente y las sumamos miembro a miembro a todas ellas, obtendremos, en virtud de que $\varphi(r) = 0$,

$$\Psi_n(r) = r^n c_n + (1 + a_1 r) r^{n-1} c_{n-1} ,$$

y de aquí

$$\Psi_{n+1}(r) = r^{n+1} c_{n+1} + (1 + a_1 r) r^n c_n ,$$
$$\Psi_{n+2}(r) = r^{n+2} c_{n+2} + (1 + a_1 r) r^{n+1} c_{n+1} ,$$

siendo claro de estas relaciones que

$$r^2 \Psi_n(r) = r^{n+2} c_n + (1 + a_1 r) r^{n+1} c_{n-1} ,$$
$$r \Psi_{n+1}(r) = r^{n+2} c_{n+1} + (1 + a_1 r) r^{n+1} c_n ,$$
$$\Psi_{n+2}(r) = r^{n+2} c_{n+2} + (1 + a_1 r) r^{n+1} c_{n+1} ,$$

o, escrito de otra forma,

$$(c_{n-1}, c_n , r^2 \Psi_n(r)) ((1 + a_1 r) r^{n+1}, r^{n+2} , - 1)^t = 0 ,$$
$$(c_n , c_{n+1}, r \Psi_{n+1}(r))((1 + a_1 r) r^{n+1}, r^{n+2} , - 1)^t = 0 ,$$
$$(c_{n+1}, c_{n+2}, \Psi_{n+2}(r)) (1 + a_1 r) r^{n+1}, r^{n+2} , - 1)^t = 0 ,$$

lo que en un espacio vectorial de dimensión 3 nos indica que los vectores de la izquierda son linealmente dependientes, es decir, que el determinante con ellos formado se anula:

$$\begin{vmatrix} c_{n-1} & c_n & r^2 \Psi_n(r) \\ c_n & c_{n+1} & r \Psi_{n+1}(r) \\ c_{n+1} & c_{n+2} & \Psi_{n+2}(r) \end{vmatrix} = 0 ;$$

y como para el presente caso se satisface una relación equivalente a la (2.6) del caso anterior, tendremos que, para n suficientemente grande,

$$\begin{vmatrix} c_{n-1} & c_n & r^2\Psi_n(r) \\ c_n & c_{n+1} & r\Psi_{n+1}(r) \\ c_{n+1} & c_{n+2} & \Psi_{n+2}(r) \end{vmatrix} \approx \Psi(r) \begin{vmatrix} c_{n-1} & c_n & r^2 \\ c_n & c_{n+1} & r \\ c_{n+1} & c_{n+2} & 1 \end{vmatrix} = 0 ,$$

o sea, que en el límite, r_1 y r_2 van a satisfacer la ecuación

$$\begin{vmatrix} c_{n-1} & c_n & z^2 \\ c_n & c_{n+1} & z \\ c_{n+1} & c_{n+2} & 1 \end{vmatrix} = 0, \qquad (2.7)$$

o equivalentemente

$$\varphi(z) \approx \frac{1}{c_{n-1}\, c_{n+1} - c_n^2} \begin{vmatrix} c_{n-1} & c_n & z^2 \\ c_n & c_{n+1} & z \\ c_{n+1} & c_{n+2} & 1 \end{vmatrix} = 0,$$

que es lo que afirma el teorema de Koenig en este segundo caso.

Pasemos ahora al caso más general y probémoslo formalmente. Es decir, supongamos que (2.1) tiene exactamente k polos, no necesariamente distintos, en el disco $|z| < R$ y que dichos polos satisfacen

$$|r_1| \leqslant |r_2| \leqslant \ldots \leqslant |r_k| < \sigma R < R ,$$

con σ un número real entre 0 y 1. Denotemos por $\varphi(z)$ al polinomio de grado k

cuyo término libre es 1 y que tiene por raíces a los k polos de F(z), esto es,

$$\varphi(z) = 1 + a_1 z + \ldots + a_k z^k = (1 - r_1^{-1} z) \ldots (1 - r_k^{-1} z), \qquad (2.8)$$

y por $\Delta(z) = 0$ a la correspondiente ecuación (2.7) en el presente caso, i.e.,

$$\Delta(z) = \begin{vmatrix} c_{n-k+1} & c_{n-k+2} & \ldots & c_n z^k \\ c_{n-k+2} & c_{n-k+3} & \ldots & c_{n+1} z^{k-1} \\ \vdots & \vdots & & \vdots \\ c_n & c_{n+1} & \ldots & c_{n+k-1} z \\ c_{n+1} & c_{n+2} & \ldots & c_{n+k} 1 \end{vmatrix} \qquad (2.9)$$

Finalmente, denotemos por $\delta(z)$ al polinomio $\Delta(z)$ normalizado (i.e., con término libre igual a la unidad). Tendremos entonces el siguiente teorema.

Teorema 2.1 (de Koenig). Bajo las hipótesis anteriores

$$\delta(z) = \varphi(z) + 0(\sigma^n), 0 < \sigma < 1 . \qquad (2.10)$$

Demostración. F(z) se puede escribir, en el caso que nos ocupa, de la siguiente manera

$$F(z) = k(z) + \frac{\Psi(z)}{\varphi(z)} , \qquad (2.11)$$

donde

$$k(z) = F(z) - \frac{\Psi(z)}{\varphi(z)} = k_0 + k_1 z + k_2 z^2 + \ldots \qquad (2.12)$$

es holomorfa en $|z| < R$ y $\Psi(z)$ es un polinomio de grado k−1 a lo más (ref 1).

Sea

$$\frac{\Psi(z)}{\varphi(z)} = \sum_{j=0}^{\infty} \gamma_j z^j \ , \qquad (2.13)$$

entonces, en virtud de que $\Psi(z)$ es un polinomio de grado $k-1$ cuando mucho, tendremos

$$\begin{aligned}
&\gamma_o a_k + \gamma_1 a_{k-1} + \ldots + \gamma_k = 0 \ , \\
&\gamma_1 a_k + \gamma_2 a_{k-1} + \ldots + \gamma_{k+1} = 0 \ , \\
&\quad \vdots \qquad\quad \vdots \qquad\qquad \vdots \qquad \vdots \\
&\gamma_n a_k + \gamma_{n+1} a_{k-1} + \ldots + \gamma_{n+k} = 0 \ , \\
&\quad \vdots \qquad\quad \vdots \qquad\qquad \vdots \qquad \vdots
\end{aligned} \qquad (2.14)$$

Sean

$$\bar{d}^{\,t}(n) = (\gamma_n, \gamma_{n+1}, \ldots, \gamma_{n+k-1}) \ ,$$

$$a^t = (a_k, a_{k-1}, \ldots, a_1) \ ;$$

de aquí y de las relaciones (2.14), es claro que

$$\bar{d}^t(n) \cdot \bar{a} = -\gamma_{n+k}, \ n = 0,1,2, \ldots ,$$

pudiéndose obtener $\bar{d}(n+1)$ a partir de $\bar{d}(n)$ como puede observarse de la siguiente relación obvia

$$\underline{X}\, \bar{d}(n) = \bar{d}(n+1), \qquad (2.15)$$

donde

$$\bar{\underline{X}} = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ -a_k & -a_{k-1} & -a_{k-2} & -a_{k-3} & \cdots & -a_1 \end{pmatrix}, \qquad (2.16)$$

es la matriz compañera del polinomio $z^k\ \varphi(z^{-1})$, i.e., la matriz cuyo polinomio característico coincide con $z^k\ \varphi(z^{-1})$, polinomio cuyas raíces son los recíprocos de los polos de F(z).

Hagamos

$$\Gamma_n = \begin{pmatrix} \gamma_{n-k+1} & \gamma_{n-k+2} & \cdots & \gamma_n \\ \gamma_{n-k+2} & \gamma_{n-k+3} & \cdots & \gamma_{n+1} \\ \vdots & \vdots & & \vdots \\ \gamma_n & \gamma_{n+1} & \cdots & \gamma_{n+k-1} \end{pmatrix}$$

$$= (d(n-k+1) \ \vdots \ d(n-k+2) \ \vdots \cdots \vdots \ \bar{d}(n)) ,$$

entonces, en virtud de (2.15), tendremos que

$$\bar{\underline{X}}\ \Gamma_n = \Gamma_{n+1} ,$$

de donde, a su vez,

$$\Gamma_n = \bar{\underline{X}}^n\ \Gamma_o , \ n = 0, 1, 2, \ldots .$$

Por otro lado, si denotamos por C_n y K_n a las matrices análogas a la matriz Γ_n pero para los coeficientes c_n y k_n, respectivamente, tendremos, por la forma en que nos viene dada F(z), que

$$C_n = \Gamma_n + K_n . \qquad (2.17)$$

Antes de proseguir, denotemos por $\| \bar{Y} \|_e$ a la norma matricial cuyo valor nos viene dado por el máximo, sobre los renglones, de las sumas de los módulos de sus elementos. Entonces, como k(z) es holomorfa en $|z| < R$, para cualquier ρ que satisfaga

$$|r_k| < \rho < R , \qquad (2.18)$$

cada elemento de K_n será $0(\rho^{-n})$ y basta con observar el último renglón de esta matriz para concluir que también

$$\| K_n \|_e = 0 (\rho^{-n}) .$$

Por otra parte, el radio espectral de $\bar{\underline{X}}^{-1}$ (esto es, el mínimo radio cuyo círculo contiene a todos los valores propios de $\bar{\underline{X}}^{-1}$) es igual a $|r_k|$; de aquí que para cualquier $\epsilon > 0$

$$\| \bar{\underline{X}}^{-1} \|_e = 0 (\epsilon + |r_k|) ,$$

(ref 6), de donde

$$\| \bar{\underline{X}}^{-n} \|_e = 0 ((\epsilon + |r_k|)^n) . \qquad (2.19)$$

Ahora bien, de (2.17) obtenemos

$$C_n \Gamma_n^{-1} = I + K_n \Gamma_n^{-1}$$

$$= I + K_n \Gamma_o^{-1} \bar{\underline{X}}^{-n} ,$$

en donde, por lo que acabamos de probar,

$$\| K_n \Gamma_o^{-1} \bar{\underline{X}}^{-n} \|_e = 0 \{\rho^{-n} (\epsilon + |r_k|)^n\} , \qquad (2.20)$$

y de estas dos últimas relaciones tendríamos que

$$\frac{\det (C_n)}{\det (\Gamma_n)} = 1 + 0 (\sigma^n) , \qquad (2.21)$$

si tan sólo pudiésemos probar que para cualquier σ que satisfaga las condiciones del teorema, es posible elegir una ρ que satisfaga (2.18) y una ϵ, de tal manera que

$$\sigma = \rho^{-1} (\epsilon + |r_k|) ,$$

lo cual se consigue fácilmente si determinamos ρ a partir de esta ecuación, es decir,

$$\rho\sigma = \epsilon + |r_k| ,$$

eligiendo la ϵ apropiadamente, lo cual, como se ve, se logra si

$$\epsilon < \sigma R - |r_k| ,$$

quedando así probada la relación (2.21).

Mediante una transformación elemental en el determinante

$$\Delta(z^{-1}) = \begin{vmatrix} c_{n-k+1} & c_{n-k+2} & \cdots & c_n & z^{-k} \\ c_{n-k+2} & c_{n-k+3} & \cdots & c_{n+1} & z^{-(k-1)} \\ \vdots & \vdots & & \vdots & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+k-1} & z^{-1} \\ c_{n+1} & c_{n+2} & \cdots & c_{n+k} & 1 \end{vmatrix} ,$$

obtenemos

$$z^k \Delta(z^{-1}) = \begin{vmatrix} z\,c_{n-k+1} - c_{n-k+2} & z\,c_{n-k+2} - c_{n-k+3} & \cdots & 0 \\ z\,c_{n-k+2} - c_{n-k+3} & z\,c_{n-k+3} - c_{n-k+4} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ z\,c_n - c_{n+1} & z\,c_{n+1} - c_{n+2} & \cdots & 0 \\ c_{n+1} & c_{n+2} & \cdots & 1 \end{vmatrix} ,$$

lo cual se reduce a

$$z^k \Delta(z^{-1}) = \det(z\,C_n - C_{n+1}) . \qquad (2.22)$$

La razón de tomar $\Delta(z^{-1})$ es que las raíces del polinomio característico de $\underline{\bar{X}}$ son los recíprocos de los polos de F(z).

Ahora podemos ya completar la demostración del teorema. Nuevamente de (2.17) tendremos que

$$z\,C_n = z\,\Gamma_n + z\,K_n ,$$

y

$$C_{n+1} = \underline{\bar{X}}\,\Gamma_n + K_{n+1} ,$$

de donde, restando miembro a miembro, tendremos

$$z\,C_n - C_{n+1} = (zI - \bar{\underline{X}})\,\Gamma_n + (z\,K_n - K_{n+1}) \ ,$$

y de aquí

$$(z\,C_n - C_{n+1})\,\Gamma_n^{-1} = (zI - \bar{\underline{X}}) + (z\,K_n - K_{n+1})\,\Gamma_o^{-1}\,\bar{\underline{X}}^{-n} \ ,$$

pudiéndose probar, de nuevo mediante (2.20), que

$$\| (z\,K_n - K_{n+1})\,\Gamma_o^{-1}\,\bar{\underline{X}}^{-n} \|_c = 0\,[\rho^{-n}(\epsilon + |r_k|)^n] \ ,$$

de donde se sigue que

$$\frac{\det\,(z\,C_n - C_{n+1})}{\det\,(\Gamma_n)\,\det\,(zI - \bar{\underline{X}})} = 1 + 0\,(\sigma^n) \ ,$$

lo que unido a (2.21) nos da

$$\frac{\det\,(z\,C_n - C_{n+1})}{\det\,(C_n)\,\det\,(zI - \bar{\underline{X}})} = 1 + 0\,(\sigma^n) \ ;$$

pero de (2.22) y sustituyendo la variable z por z^{-1} , tendremos que

$$\frac{z^{-k}\,\Delta\,(z)}{\det\,(C_n)\,z^{-k}\,\varphi(z)} = \frac{\Delta\,(z)}{\det\,(C_n)\,\varphi(z)} = 1 + 0\,(\sigma^n),$$

esto es,

$$\frac{\Delta(z)}{\det\ (C_n)} \;=\; \varphi(z) \;+\; 0\,(\sigma^n),$$

y como se puede apreciar de (2.9), det (C_n) es precisamente el término libre del polinomio $\Delta(z)$, de tal manera que el miembro ezquierdo de esta expresión no es más que el polinomio normalizado $\delta(z)$, es decir,

$$\delta(z) \;=\; \varphi(z) \;+\; 0\ (\sigma^n)\,,$$

con lo que se termina la demostración del teorema de Koenig.

De este teorema se deriva un corolario obvio pero muy importante. Para ello, debemos observar que tanto el término de la potencia mayor de z como el término independiente del polinomio $\Delta(z)$, pueden obtenerse inmediatamente, es decir,

$$\Delta(z) = (-1)^k\ \det\ (C_{n+1})\, z^k + \ldots + \det\,(C_n).$$

Si denotamos a det (C_n) por $H_{n,k}$, i.e.,

$$H_{n,\,k} = \begin{vmatrix} c_{n-k+1} & c_{n-k+2} & \cdots & c_n \\ c_{n-k+2} & c_{n-k+3} & \cdots & c_{n+1} \\ \vdots & \vdots & & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+k-1} \end{vmatrix}, \qquad (2.23)$$

donde el subíndice k nos indica el orden del determinante, el corolario nos diría lo siguiente.

Corolario 2.2 (al teorema de Koenig). Bajo las mismas hipótesis de dicho teorema

$$\frac{H_{n,k}}{H_{n+1,k}} = r_1 \; r_2 \; \ldots \; r_k \; + \; 0(\sigma^n). \qquad (2.24)$$

Nota.- A estas $H_{n,k}$ son a las que se les conoce por el nombre de determinantes de Haenkel.

2.1.2 Motivación del Algoritmo Cociente–Diferencia (Q–D). Los determinantes de Haenkel son de gran importancia en la construcción del modelo Cociente-Diferencia. Su utilidad nos viene dada básicamente por una fórmula de recurrencia debida a Aitken y de fácil deducción.

Sabemos ya que

$$H_{n,2} = \begin{vmatrix} c_{n-1} & c_n \\ c_n & c_{n+1} \end{vmatrix} = c_{n+1} \; c_{n-1} \; - \; c_n^2 \; ,$$

pero $c_n = H_{n,1}$, de donde

$$H_{n,2} = H_{n+1,1} \; H_{n-1,1} \; - \; H_{n,1}^2 \; .$$

Esto nos sugiere tomar

$$d\,H_{n,3} = H_{n+1,2} \; H_{n-1,2} \; - \; H_{n,2}^2 \; ,$$

donde d es una constante por determinarse.

Por un lado tenemos que

$$d\,H_{n,3} = d \begin{vmatrix} c_{n-2} & c_{n-1} & c_n \\ c_{n-1} & c_n & c_{n+1} \\ c_n & c_{n+1} & c_{n+2} \end{vmatrix}$$

$$= d\,(c_{n-2}\,c_n\,c_{n+2} - c_{n-2}\,c_{n+1}^2 - c_{n-1}^2\,c_{n+2} + 2\,c_{n-1}\,c_n\,c_{n+1} + - c_n^3)\ ;$$

por otra parte

$$H_{n+1,2}\,H_{n-1,2} - H_{n,2}^2 = (c_n\,c_{n+2} - c_{n+1}^2)(c_{n-2}\,c_n - c_{n-1}^2) - (c_{n-1}\,c_{n+1} - c_n^2)^2$$

$$= c_{n-2}\,c_n^2\,c_{n+2} - c_{n-2}\,c_n\,c_{n+1}^2 - c_{n-1}^2\,c_n\,c_{n+2} + c_{n-1}^2\,c_{n+1}^2 + - c_{n-1}^2\,c_{n+1}^2 + 2\,c_{n-1}\,c_n^2\,c_{n+1} - c_n^4$$

$$= c_n\,(c_{n-2}\,c_n\,c_{n+2} - c_{n-2}\,c_{n+1}^2 - c_{n-1}^2\,c_{n+2} + 2\,c_{n-1}\,c_n\,c_{n+1} + - c_n^3)\ ;$$

de donde, si hacemos $d = c_n = H_{n,1}$, tendremos que

$$H_{n,1}\,H_{n,3} = H_{n+1,2}\,H_{n-1,2} - H_{n,2}^2\ ,$$

siendo fácil probar (por inducción, por ejemplo) la fórmula general

$$H_{n,k-1}\,H_{n,k+1} = H_{n+1,k}\,H_{n-1,k} - H_{n,k}^2\ , \qquad (2.25)$$

que es la requerida fórmula de recurrencia de Aitken.

Esta fórmula nos permite el cálculo recursivo de los determinantes $H_{n,k}$ de Haenkel a partir de las c_j, lo cual se ve claro si construimos la siguiente tabla para dichos determinantes

TABLA 2.1. TABLA DE DETERMINANTES DE HAENKEL

1					
1	$H_{0,1}$				
1	$H_{1,1}$	$H_{0,2}$			
1	$H_{2,1}$	$H_{1,2}$	$H_{0,3}$		
1	$H_{3,1}$	$H_{2,2}$	$H_{1,3}$	$H_{0,4}$	
1	$H_{4,1}$	$H_{3,2}$	$H_{2,3}$	$H_{1,4}$	$H_{0,5}$
⋮	⋮	⋮	⋮	⋮	⋮

Debemos notar que además de conocer la primera columna de esta tabla, pues definimos $H_{n,0} = 1$, conocemos también la segunda ya que $H_{n,1} = c_n$. Entonces la fórmula de Aitken se puede expresar mnemotécnicamente si hacemos que $H_{n,k}$ sea el elemento central de dicha tabla y lo denotamos por C y denotamos a los elementos vecinos por puntos cardinales, es decir, N, S, NO, SE, etc. Así, la fórmula de Aitken nos vendría dada por

$$C^2 = NS - NO\ (SE). \qquad (2.26)$$

Sin embargo, la utilización de esta fórmula para generar la tabla 2.1 requiere de muchos cálculos y, por otro lado, la utilidad de la misma es manifiesta pues, por el Corolario 2.2, los cocientes de elementos sucesivos en la columna j tienden al producto de los j primeros polos de F(z) cuando $n \to \infty$. Por lo tanto, sería conveniente tener un modelo equivalente pero mucho menos laborioso en su formación.

Para ello, démonos cuenta que el único interés que para nosotros guardan los determinantes $H_{n,k}$ está en relación con las cantidades

$$q_{nk} = \frac{H_{n+1,k}\ H_{n-1,k-1}}{H_{n,k}\ H_{n,k-1}}, \qquad (2.27)$$

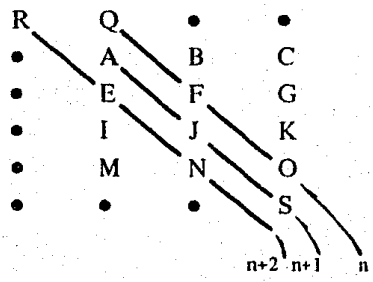
pues cuando $n \to \infty$, $q_{nk} \to r_k^{-1}$, suponiendo que se satisface la condición

$$|r_1| < |r_2| < \ldots < |r_k|.$$

La razón de definir a las q_{nk} de tal manera que tiendan a los recíprocos de los polos de F(z) radica en el hecho de que mediante ellas se obtendrán relaciones muy importantes que de otra manera no serían factibles, como se verá más adelante.

Estas cantidades surgen de manera natural a partir de la expansión en fracciones continuas de la función F(z) (refs 7, 15). Sin embargo, existe una manera de motivar la selección de ciertas cantidades análogas, que denotaremos por e_{nk}, a partir tan sólo de las primeras y de la fórmula (2.26) de Aitken, quedándonos al final un par de importantes fórmulas recursivas tanto para q_{nk} como para e_{nk} con las que generaremos el modelo equivalente que requerimos. Veamos cómo es esto.

Como lo que buscamos es recursión lo natural es tratar de encontrar una fórmula que nos relacione q_{nk} con $q_{n+1,k}$. A continuación se muestra la forma en que la relación (2.26) puede ser aprovechada para conseguir esto. Para ello, consideremos la siguiente porción de la tabla 2.1, suponiendo que ésta se halla centrada en $F = H_{n,k}$.



(2.28)

Así pues,

$$q_{nk} = \frac{H_{n+1,k}\ H_{n-1,k-1}}{H_{nk}\ H_{n,k-1}} = \frac{JQ}{FA} = \frac{QJ}{AF}\ ,$$

$$q_{n+1,k} = \frac{H_{n+2,k}\ H_{n,k-1}}{H_{n+1,k}\ H_{n+1,k-1}} = \frac{NA}{JE} = \frac{AN}{EJ}, \qquad (2.29)$$

de tal manera que si queremos relacionar a estos dos elementos tendremos que aplicar dos veces la fórmula (2.26) tomando como elementos centrales a A y a J, respectivamente,

$$A^2 = QE - RF \quad ; \qquad J^2 = FN - OE \quad ;$$

y enseguida forzar el que los primeros términos de los segundos miembros de estas dos relaciones sean las correspondientes cantidades (2.29); esto se consigue de la forma siguiente:

$$\frac{AJ}{EF} = \frac{QJ}{AF} - \frac{JR}{AE} \quad ; \quad \frac{AJ}{EF} = \frac{AN}{EJ} - \frac{OA}{FJ}\ ,$$

con lo que no sólo conseguimos eso, sino, lo más importante, que ambas relaciones sean una misma cosa y podamos por tanto igualarlas; pero antes conviene observar que los segundos términos en los miembros derechos de estas igualdades son cantidades análogas a los q_{nk}, como puede observarse de (2.28), por lo que si hacemos

$$\frac{OA}{FJ} = \frac{H_{n+1,k+1}\ H_{n,k-1}}{H_{n,k}\ H_{n+1,k}} = e_{nk}\ , \qquad (2.30)$$

tendremos entonces que

$$\frac{JR}{AE} = \frac{H_{n+1,k}\ H_{n,k-2}}{H_{n,k-1}\ H_{n+1,k-1}} = e_{n,k-1}\ ;$$

teniendo así que, igualando y trasponiendo términos en las últimas ecuaciones,

$$q_{nk} + c_{nk} = q_{n+1,k} + e_{n,k-1}\ . \qquad (2.31)$$

Pero necesitamos otra relación para poder determinar todos los elementos del modelo que se quiere construir a partir de $q_{n1} = c_{n+1}/c_n$. Dicha relación se obtiene inmediatamente con tan sólo observar que

$$q_{nk} \cdot e_{nk} = \frac{QJ}{AF} \cdot \frac{OA}{FJ} = \frac{OQ}{F^2} = \frac{H_{n+1,k+1}\ H_{n-1,k-1}}{H^2_{n,k}},$$

$$q_{n+1,k} \cdot e_{n,k-1} = \frac{AN}{EJ} \cdot \frac{JR}{AE} = \frac{NR}{E^2} = \frac{H_{n+2,k}\ H_{n,k-2}}{H^2_{n+1,k-1}}\ ;$$

esto es, que ambas cantidades son análogas, únicamente se encuentran localizadas en diagonales distintas (ver esquema (2.28)), digamos n y n+2, respectivamente, de tal manera que si queremos que las dos sean iguales solamente tenemos que sustituir en la segunda relación n por n-1 y k por k+1, de donde obtendríamos la fórmula requerida

$$q_{nk} \cdot e_{nk} = q_{n,k+1} \cdot e_{n-1,k}\ . \qquad (2.32)$$

En virtud de que la relación (2.31) nos da las e_{nk} en términos de las diferencias de los cocientes q_{nk}, a este método se le conoce con el nombre de Algoritmo Cociente-Diferencia o Algoritmo Q–D (quotient-difference). Mediante el mismo se puede construir una tabla análoga a la construida para los determinantes de Haenkel, pero mucho más fácil de calcular mediante las fórmulas (2.31) y

(2.32). (Tabla 2.2). Y análogamente a como ocurrió con los determinantes de Haenkel, en esta tabla conocemos las dos primeras columnas ya que, por un lado, definimos $e_{n0} = 0$ y, por el otro, tenemos que $q_{n1} = c_{n+1}/c_n$, ambas para $n \geq 0$. Como se muestra en la tabla 2.2, esto es

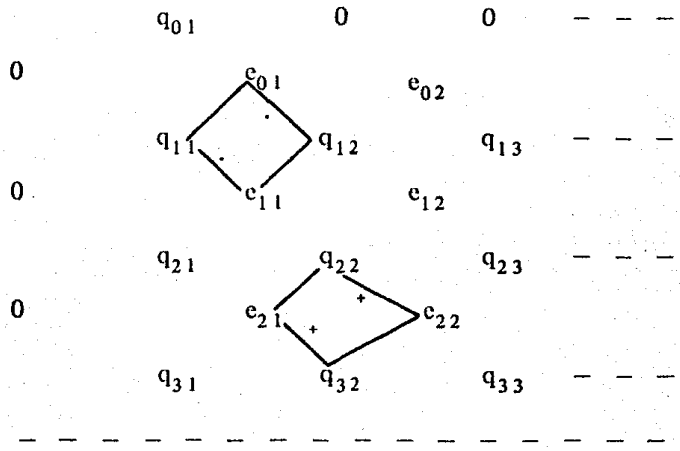


TABLA 2.2. MODELO COCIENTE–DIFERENCIA (Q–D)

suficiente para generar el modelo completo mediante las relaciones (2.31) y (2.32), las cuales, por otra parte, nos relacionan, cada una por su lado, cuatro elementos de la tabla 2.2 localizados en los vértices de un rombo; de aquí que a dichas fórmulas se les conozca también con el nombre de reglas del rombo.

Finalmente, y al igual que ocurre con la tabla 2.1 existen dos maneras de generar el modelo Q–D. La primera es utilizando la fórmula (2.31) para calcular las e_{nk} y la (2.32) para las q_{nk}. Esto nos genera la tabla 2.2 columna por columna y presenta el siguiente riesgo: como bajo la suposicón $|r_1| < |r_2| < \ldots < |r_k|$ tendríamos que $q_{n1} \rightarrow r_1^{-1}$, cuando $n \rightarrow \infty$, entonces al calcular e_{n1} mediante la fórmula

$$e_{n1} = q_{n+1,1} - q_{n1} \ ,$$

tendríamos una cantidad que tendería a cero, perdiendo así exactitud al calcular q_{nk} mediante la fórmula

$$q_{n2} = \frac{e_{n1}}{e_{n-1,1}} \; q_{n1} \; .$$

Es decir, tendríamos un método inestable.

La otra manera de generar la tabla 2.2 es por renglones utilizando la fórmula (2.31) para calcular las q_{nk} y la (2.32) las e_{nk}. Así, al calcular las q_{nk} mediante la igualdad.

$$q_{n+1,k} = q_{nk} + e_{nk} - e_{n,k-1} \; ,$$

podremos tener pérdida de exactitud en los primeros pasos al toparnos con valores grandes de las e_{nk}, pero al final el método resulta mucho más estable ya que bajo la suposición de módulos distintos tanto e_{nk} como $e_{n,k-1}$ tenderán a cero cuando $n \to \infty$.

Observación.- Hasta ahora no habíamos hecho ninguna consideración sobre la necesidad de que los determinantes $H_{n,k}$ sean distintos de cero para no perder continuidad en la motivación. Sin embargo, puede probarse que para n suficientemente grande, $H_{n,k} \neq 0$ bajo las hipótesis que hemos estado manejando (refs 7, 14)

2.2 LA TABLA DE PADE Y EL ALGORITMO COCIENTE–DIFERENCIA

2.2.1 Aproximación de la Función $F(z) = \sum_{j=0}^{\infty} c_j z^j$ Mediante la Función Racional $A_{n,k}(z)/B_{n,k}(z)$, donde $A_{n,k}(z)$ es un Polinomio de Grado n y $B_{n,k}(z)$ un Polinomio de Grado k.- La Tabla de Padé.

Consideremos nuevamente a la serie de potencias

$$F(z) = \sum_{j=0}^{\infty} c_j z^j \ , \ c_o \neq 0 . \qquad (2.33)$$

Lo que queremos ahora es construir una función racional $A_{n,k}(z)/B_{n,k}(z)$ – con $A_{n,k}$ y $B_{n,k}$ polinomios de grados n y k, respectivamente– tal que

$$F(z) - \frac{A_{n,k}(z)}{B_{n,k}(z)} = \frac{B_{n,k}(z)\,F(z) - A_{n,k}(z)}{B_{n,k}(z)} = \frac{\sum_{j=n+k+1}^{\infty} c_j' z^j}{B_{n,k}(z)} = O(z^{n+k+1}) \ ,$$

donde $A_{n,k}$ y $B_{n,k}$ están unívocamente determinados y $A_{n,k}$ es tal que anula los primeros n términos de $B_{n,k}(z)\,F(z)$ y $B_{n,k}$ los siguientes k términos. Veamos de que forma tienen que ser $A_{n,k}$ y $B_{n,k}$ para que esto se cumpla.

En primer lugar, si hacemos

$$B_{n,k}(z)\,F(z) = \sum_{j=0}^{\infty} d_j z^j \ , \qquad (2.34)$$

tendremos, por lo anterior, que

$$d_j = 0, \ j = n+1, \ n+2, \ \ldots \ , \ n+k \ ; \qquad (2.35)$$

por otra parte, si

$$B_{n,k}(z) = \sum_{j=0}^{k} t_j z^j \ , \qquad (2.36)$$

es claro que

$$B_{n,k}(z)\,F(z) = c_0 t_0 + (c_1 t_0 + c_o t_1)\, z + (c_2 t_0 + c_1 t_1 + c_0 t_2)\, z^2 + \ldots$$
$$= t_0\,(c_0 + c_1 z + \ldots) + t_1 z\,(c_0 + c_1 z + \ldots) + \ldots \ ,$$

de tal manera que si hacemos

$$F_l(z) = \sum_{j=0}^{l} c_j z^j , \qquad (2.37)$$

$$D_l(z) = \sum_{j=0}^{l} d_j z^j , \qquad (2.38)$$

tendremos que

$$D_{n+k}(z) = \sum_{j=0}^{n+k} d_j z^j$$

$$= t_0 F_{n+k}(z) + t_1 z F_{n+k-1}(z) + \ldots + t_k z^k F_n(z), \qquad (2.39)$$

y, en virtud de (2.35),

$$D_l(z) = D_n(z), \quad l = n+1, n+2, \ldots, n+k , \qquad (2.40)$$

donde, debido a la suposición que hicimos sobre $A_{n,k}(z)$, es claro que

$$D_n(z) = A_{n,k}(z) . \qquad (2.41)$$

Es decir, la suposición (2.35) nos lleva a la resolución del sistema

$$\begin{aligned}
&t_0 F_n(z) + t_1 z F_{n-1}(z) + \ldots + t_k z^k F_{n-k}(z) = D_n(z) = A_{n,k}(z) \\
&t_0 F_{n+1}(z) + t_1 z F_n(z) + \ldots + t_k z^k F_{n-k+1}(z) = A_{n,k}(z) \\
&\quad\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
&t_0 F_{n+k-1}(z) + t_1 z F_{n+k-2}(z) + \ldots + t_k z^k F_{n-1}(z) = A_{n,k}(z) \\
&t_0 F_{n+k}(z) + t_1 z F_{n+k-1}(z) + \ldots + t_k z^k F_n(z) = A_{n,k}(z) ,
\end{aligned} \qquad (2.42)$$

para determinar la forma de $B_{n,k}$, i.e., éste es un sistema de k+1 ecuaciones en las k+1 incógnitas $t_0, t_1, \ldots, t_k$.

Las siguientes manipulaciones algebráicas con las ecuaciones del sistema (2.42)

a) sustituir la i-ésima ecuación por lo que resulte de restar a la ecuación (i + 1) –ésima la ecuación i-ésima, i = 1, 2, . . . , k ;

b) en el sistema resultante de la operación anterior, sustituir la ecuación k + 1 por lo que resulte de restarle a ésta la suma de todas las anteriores,

nos llevan al sistema equivalente

$$
\begin{aligned}
& c_{n+1} z^{n+1} t_o + c_n z^{n+1} t_1 + \ldots + c_{n-k+1} z^{n+1} t_k = 0 \\
& c_{n+2} z^{n+2} t_o + c_{n+1} z^{n+2} t_1 + \ldots + c_{n-k+2} z^{n+2} t_k = 0 \\
& \quad \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \qquad\quad \vdots \qquad ; \\
& c_{n+k} z^{n+k} t_o + c_{n+k-1} z^{n+k} t_1 + \ldots + c_n z^{n+k} t_k = 0 \\
& F_n(z)\, t_o + z F_{n-1}(z)\, t_1 + \ldots + z^k F_{n-k}(z) t_k = A_{n,k}(z)
\end{aligned}
$$

pero como de aquí

$$
\begin{aligned}
& z^{n+1} (c_{n-k+1} t_k + c_{n-k+2} t_{k-1} + \ldots + c_{n+1} t_o) = 0 \\
& z^{n+2} (c_{n-k+2} t_k + c_{n-k+3} t_{k-1} + \ldots + c_{n+2} t_0) = 0 \\
& \quad \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \qquad\quad \vdots \qquad , \\
& z^{n+k} (c_n t_k + c_{n+1} t_{k-1} + \ldots + c_{n+k} t_0) = 0 \\
& z^k F_{n-k}(z)\, t_k + z^{k-1} F_{n-k+1}(z) t_{k-1} + \ldots + F_n(z) t_0 = A_{n,k}(z)
\end{aligned}
$$

estro sistema original es enteramente equivalente al sistema

$$\begin{aligned}
&c_{n-k+1}\, t_k + c_{n-k+2}\, t_{k-1} + \cdots + c_{n+1}\, t_0 = 0\\
&c_{n-k+2}\, t_k + c_{n-k+3}\, t_{k-1} + \cdots + c_{n+2}\, t_0 = 0\\
&\quad\vdots \qquad\qquad \vdots \qquad\qquad\quad \vdots \qquad \vdots \qquad ; \qquad (2.43)\\
&c_n\, t_k + c_{n+1}\, t_{k-1} + \cdots + c_{n+k}\, t_0 = 0\\
&z^k F_{n-k}(z) t_k + z^{k-1} F_{n-k+1}(z) t_{k-1} + \cdots + F_n(z) t_0 = A_{n,k}(z)
\end{aligned}$$

este sistema se resuelve en términos del determinante

$$\begin{vmatrix} c_{n-k+1} & c_{n-k+2} & \cdots & c_{n+1} \\ c_{n-k+2} & c_{n-k+3} & \cdots & c_{n+2} \\ \vdots & \vdots & & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+k} \\ z^k F_{n-k}(z) & z^{k-1} F_{n-k+1}(z) & \cdots & F_n(z) \end{vmatrix} = \begin{vmatrix} c_{n-k+1} & c_{n-k+2} & \cdots & z^k F_{n-k}(z) \\ c_{n-k+2} & c_{n-k+3} & \cdots & z^{k-1} F_{n-k+1}(z) \\ \vdots & \vdots & & \vdots \\ c_{n+1} & c_{n+2} & \cdots & F_n(z) \end{vmatrix} = N_{n,k}(z), \quad (2.44)$$

e la siguiente manera

$$t_i = \frac{\begin{vmatrix} c_{n-k+1} & \cdots & c_{n-i} & 0 & c_{n-i+2} & \cdots & c_{n+1} \\ c_{n-k+2} & \cdots & c_{n-i+1} & 0 & c_{n-i+3} & \cdots & c_{n+2} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ c_n & \cdots & c_{n+k-i} & 0 & c_{n+k-i+2} & \cdots & c_{n+k} \\ z^k F_{n-k}(z) & \cdots & & A_{n,k}(z) & & \cdots & F_n(z) \end{vmatrix}}{N_{n,k}(z)},$$

de aquí

$$t_i = \frac{A_{n,k}(z)}{N_{n,k}(z)} \cdot M_i \,, \quad i = 0, 1, \ldots, k \,, \qquad (2.45)$$

donde

$$M_i = \begin{vmatrix} c_{n-k+1} & \cdots & c_{n-i} & c_{n-i+2} & \cdots & c_{n+1} \\ c_{n-k+2} & \cdots & c_{n-i+1} & c_{n-i+3} & \cdots & c_{n+2} \\ \vdots & & \vdots & \vdots & & \vdots \\ c_n & \cdots & c_{n+k-i} & c_{n+k-i+2} & \cdots & c_{n+k} \end{vmatrix} \qquad (2.46)$$

es el menor correspondiente a la i-ésima columna del determinante original. Pero t_i es el i-ésimo coeficiente del polinomio $B_{n,k}(z)$, es decir, una constante, de la misma forma en que de (2.46) se ve que M_i es independiente de z, de tal manera que para que (2.45) se satisfaga debemos tener que

$$A_{n,k}(z) = c \cdot N_{n,k}(z) \,, \qquad (2.47)$$

c constante, y de aquí, por (2.45),

$$t_i = c \cdot M_i \,, \qquad (2.48)$$

para toda i, con c la misma constante.

Tratemos de deducir ahora una fórmula análoga para el polinomio $B_{n,k}(z)$. Sabemos que

$$B_{n,k}(z) = t_0 + t_1 z + \ldots + t_{k-1} z^{k-1} + t_k z^k \ ,$$

de donde, utilizando la relación (2.48) anterior, tendremos que

$$\begin{aligned} B_{n,k}(z) &= c\,(M_0 + M_1 z + \ldots + M_{k-1} z^{k-1} + M_k z^k) \\ &= c\,K_{n,k}(z)\ , \end{aligned} \qquad (2.49)$$

donde

$$K_{n,k}(z) = \begin{vmatrix} c_{n-k+1} & c_{n-k+2} & \cdots & c_n & z^k \\ c_{n-k+2} & c_{n-k+3} & \cdots & c_{n+1} & z^{k-1} \\ \vdots & \vdots & & \vdots & \vdots \\ c_{n+1} & c_{n+2} & \cdots & c_{n+k} & 1 \end{vmatrix} \ . \qquad (2.50)$$

Conviene señalar, por otro lado, que este determinante $K_{n,k}(z)$ no es otro que el determinante $\Delta(z)$ utilizado en la demostración del teorema de Koenig en la primera sección, y que, en virtud de las relaciones (2.47) y (2.49), se tendrá la importante fórmula

$$\frac{A_{n,k}(z)}{B_{n,k}(z)} = \frac{N_{n,k}(z)}{K_{n,k}(z)} \ , \qquad (2.51)$$

de donde, por el mismo teorema de Koenig, dicha igualdad entre determinantes resulta clara ya que $A_{n,k}(z)/B_{n,k}(z)$ es una aproximación a la función $F(z)$. En lo sucesivo, se irán aclarando otra serie de puntos al surgir similitudes igualmente interesantes.

Además, con esos cocientes se podrá construir una tabla doblemente infinita en la que el elemento (n, k) vendrá dado por la aproximación $N_{n,k}(z)/K_{n,k}(z)$, como se muestra enseguida.

n \ k	1	2	3		— —	— —		k	— —
0	$\frac{1}{K_{01}(z)}$	$\frac{1}{K_{02}(z)}$	$\frac{1}{K_{03}(z)}$		— —	— —		$\frac{1}{K_{0k}(z)}$	— —
1	$\frac{N_{11}(z)}{K_{11}(z)}$	$\frac{N_{12}(z)}{K_{12}(z)}$	$\frac{N_{13}(z)}{K_{13}(z)}$		— —	— —		$\frac{N_{1k}(z)}{K_{1k}(z)}$	— —
2	$\frac{N_{21}(z)}{K_{21}(z)}$	$\frac{N_{22}(z)}{K_{22}(z)}$	$\frac{N_{23}(z)}{K_{23}(z)}$		— —	— —		$\frac{N_{2k}(z)}{K_{2k}(z)}$	— —
⋮	⋮	⋮	⋮					⋮	
n	$\frac{N_{n1}(z)}{K_{n1}(z)}$	$\frac{N_{n2}(z)}{K_{n2}(z)}$	$\frac{N_{n3}(z)}{K_{n3}(z)}$		— —	— —		$\frac{N_{nk}(z)}{K_{nk}(z)}$	— —
⋮	⋮	⋮	⋮					⋮	
⋮	⋮	⋮	⋮					⋮	

TABLA 2.3. TABLA DE PADE

Esta tabla es la llamada Tabla de Padé y es interesnate observar que cuando F(z) es la función racional Q(z)/P(z), con Q y P polinomios de grados m y n, respectivamente, entonces el elemento (m, n) de la tabla de Padé para F(z) nos viene dado por Q(z)/P(z) precisamente.

Pero para que la relación (2.51) resulte realmente útil , sería bueno que tuviésemos fórmulas de recurrencia para $N_{n,k}(z)$ y $K_{n,k}(z)$. Se darán en el próximo párrafo unas cuantas de ellas que valen tanto para uno como para el otro polinomio.

2.2.2 Obtención del Modelo Q–D a Partir de la Tabla de Padé. Consideremos al determinante $K_{n,k}(z)$. Las fórmulas que se deducirán ahora para este polinomio son igualmente aplicables al determinante $N_{n,k}(z)$ y la forma de hacerlo es enteramente análoga. La única razón de hacerlo para $K_{n,k}(z)$ en vez de $N_{n,k}(z)$ es que para aquél la notación resulta un poco menos engorrosa.

Sea $k = 1$, entonces

$$K_{n,1}(z) = \begin{vmatrix} c_n & z \\ c_{n+1} & 1 \end{vmatrix} = c_n - z\,c_{n+1} ,$$

$$K_{n+1,1}(z) = \begin{vmatrix} c_{n+1} & z \\ c_{n+2} & 1 \end{vmatrix} = c_{n+1} - z\,c_{n+2} ,$$

de donde, multiplicando estas ecuaciones por c_{n+1} y c_n , respectivamente,

$$c_{n+1}\quad K_{n,1}(z) = c_{n+1}\,c_n - z\,c_{n+1}^2 ,$$

$$c_n\quad K_{n+1,1}(z) = c_n\,c_{n+1} - z\,c_n\,c_{n+2} ,$$

y de aquí, restando miembro a miembro la segunda de la primera,

$$c_{n+1} K_{n,1}(z) - c_n K_{n+1,1}(z) = z\,(c_n\, c_{n+2} - c_{n+1}^2),$$

o lo que es lo mismo

$$H_{n+1,1} K_{n,1}(z) = H_{n,1}\; K_{n+1,1}(z) + z\, H_{n+1,2}\ .$$

Esto nos sugiere tomar, en el caso k = 2,

$$H_{n+1,2} K_{n,2}(z) = H_{n,2}\; K_{n+1,2}(z) + z\, H_{n+1,3}\; d(z)\ ,$$

donde d(z) está por determinarse.

Por un lado

$$H_{n+1,2} K_{n,2}\,(z) = \begin{vmatrix} c_n & c_{n+1} \\ c_{n+1} & c_{n+2} \end{vmatrix} \begin{vmatrix} c_{n-1} & c_n & z^2 \\ c_n & c_{n+1} & z \\ c_{n+1} & c_{n+2} & 1 \end{vmatrix}$$

$$= (c_n c_{n+2} - c_{n+1}^2)\,[(c_n c_{n+2} - c_{n+1}^2) z^2 - (c_{n-1} c_{n+2} - c_n\, c_{n+1})\, z + c_{n-1} c_{n+1} - c_n^2\,]$$

$$= (c_n^2 c_{n+2}^2 - 2\, c_n c_{n+1}^2 c_{n+2} + c_{n+1}^4)\, z^2 - (c_{n-1} c_n c_{n+2}^2 + - c_{n-1} c_{n+1}^2\, c_{n+2} - c_n^2 c_{n+1} c_{n+2} + c_n\, c_{n+1}^3)\, z + + (c_n c_{n+2} - c_{n+1}^2)\,(c_{n-1}\, c_{n+1} - c_n^2)\,;$$

y por otra parte,

$$H_{n,2}\, K_{n+1,2}(z) + z\, H_{n+1,3}\, d(z) =$$

$$= \begin{vmatrix} c_{n-1} & c_n \\ c_n & c_{n+1} \end{vmatrix} \begin{vmatrix} c_n & c_{n+1} & z^2 \\ c_{n+1} & c_{n+2} & z \\ c_{n+2} & c_{n+3} & 1 \end{vmatrix} + z \begin{vmatrix} c_{n-1} & c_n & c_{n+1} \\ c_n & c_{n+1} & c_{n+2} \\ c_{n+1} & c_{n+2} & c_{n+3} \end{vmatrix} \quad d(z)$$

$$= (c_{n-1}c_{n+1} - c_n^2)\,[(c_{n+1}c_{n+3} - c_{n+2}^2)\,z^2 - (c_n c_{n+3} - c_{n+1}c_{n+2})\,z +$$

$$+ (c_n c_{n+2} - c_{n+1}^2)\,] +$$

$$+ (c_{n-1}c_{n+1}c_{n+3} - c_{n-1}c_{n+2}^2 - c_n^2 c_{n+3} + c_n c_{n+1} c_{n+2} +$$

$$+ c_n c_{n+1} c_{n+2} - c_{n+1}^3)\, z \cdot d(z)$$

$$= (c_{n-1}c_{n+1}^2 c_{n+3} - c_n^2 c_{n+1} c_{n+3} - c_{n-1}c_{n+1}c_{n+2}^2 + c_n^2 c_{n+2}^2)\, z^2 +$$

$$- (c_{n-1}c_n c_{n+1} c_{n+3} - c_n^3 c_{n+3} - c_{n-1}c_{n+1}^2 c_{n+2} + c_n^2 c_{n+1} c_{n+2})\, z +$$

$$+ (c_{n-1}c_{n+1}c_{n+3} - c_{n-1}c_{n+2}^2 - c_n^2 c_{n+3} + 2\, c_n c_{n+1} c_{n+2} +$$

$$- c_{n+1}^3)\, z \cdot d(z) +$$

$$+ (c_{n-1}c_{n+1} - c_n^2)\,(c_n c_{n+2} - c_{n+1}^2)\,.$$

Si tomamos $d(z) = K_{n,1}(z) = \begin{vmatrix} c_n & z \\ c_{n+1} & 1 \end{vmatrix}$, tendremos que

$$H_{n,2}\ K_{n+1,2}(z) + z\,H_{n+1,3}\,d(z) =$$

$$= (c_n^2\,c_{n+2}^2 - 2\,c_n c_{n+1}^2\,c_{n+2} + c_{n+1}^4)\,z^2 - (c_{n-1}c_n\,c_{n+2}^2 - c_n^2 c_{n+1}\,c_{n+2} +$$

$$+ c_n c_{n+1}^3 - c_{n-1}\,c_{n+1}^2\,c_{n+2})\,z + (c_{n-1}c_{n+1} - c_n^2)\,(c_n c_{n+2} - c_{n+1}^2)\,,$$

de donde

$$H_{n+1,2}\,K_{n,2}\,(z) = H_{n,2}\,K_{n+1,2}(z) + z\,H_{n+1,3}\,K_{n,1}(z)\,,$$

pudiéndose probar (por inducción, por ejemplo) la fórmula general

$$H_{n+1,k}\,P_{n,k}(z) = H_{n,k}P_{n+1,k}(z) + z\,H_{n+1,k+1}\,P_{n,k-1}(z)\,, \qquad (2.52)$$

donde $P_{n,k}(z)$ denota a cualquiera de los polinomios $N_{n,k}(z)$, $K_{n,k}(z)$. De una manera enteramente similar puede llegarse a una segunda fórmula de recursión para los mismos polinomios:

$$H_{n,k+1}P_{n,k}(z) = H_{n,k}\,P_{n,k+1}(z) + z\,H_{n+1,k+1}\,P_{n-1,k}(z). \qquad (2.53)$$

Por otro lado, sucede que en la práctica es generalmente más conveniente trabajar o bien con polinomios mónicos, o bien con aquéllos que tienen su término libre igual a la unidad. Si queremos esta última normalización para los polinomios $N_{n,k}(z)$ y $K_{n,k}(z)$, es fácil ver que tenemos que hacer las siguientes sencillas transformaciones

$$n_{n,k}(z) = \frac{N_{n,k}(z)}{H_{n,k}}\ ,\ k_{n,k}(z) = \frac{K_{n,k}\,(z)}{H_{n,k}}\ , \qquad (2.54)$$

es decir,

$$p_{n,k}(z) = \frac{P_{n,k}(z)}{H_{n,k}} , \qquad (2.55)$$

si nuevamente denotamos por $p_{n,k}(z)$ a cualquiera de los polinomios $n_{n,k}(z)$, $k_{n,k}(z)$. Sustituyendo $P_{n,k}(z) = H_{n,k} p_{n,k}(z)$ en (2.52), obtenemos

$$H_{n,k} H_{n+1,k} p_{n,k}(z) = H_{n,k} H_{n+1,k} p_{n+1,k}(z) + z H_{n+1,k+1} H_{n,k-1} p_{n,k-1}(z),$$

$$p_{n,k}(z) = p_{n+1,k}(z) + e_{nk} z p_{n,k-1}(z), \qquad (2.56)$$

donde definimos

$$e_{nk} = \frac{H_{n+1,k+1} H_{n,k-1}}{H_{n,k} H_{n+1,k}} . \qquad (2.57)$$

La misma sustitución, pero en (2.53), nos lleva a lo siguiente

$$H_{n,k+1} H_{n,k} p_{n,k}(z) = H_{n,k+1} H_{n,k} p_{n,k+1}(z) + z H_{n+1,k+1} H_{n-1,k} p_{n-1,k}(z),$$

$$p_{n,k}(z) = p_{n,k+1}(z) + q_{n,k+1} z p_{n-1,k}(z) , \qquad (2.58)$$

donde hicimos

$$q_{nk} = \frac{H_{n+1,k} H_{n-1,k-1}}{H_{n,k} H_{n,k-1}} . \qquad (2.59)$$

Es interesante observar que las cantidades e_{nk}, q_{nk} que hemos definido aquí no son otras que las que se motivaron en la primera sección de este capítulo.

Otras dos fórmulas de recurrencia, similares a las dadas por (2.52) y (2.53), pueden deducirse de manera análoga a como se hizo con estas últimas:

$$H_{n+1,k}P_{n,k+1}(z) = H_{n,k+1}P_{n+1,k}(z) - z\,H_{n+1,k+1}P_{n,k}(z), \qquad (2.60)$$

$$H_{n+1,k}P_{n-1,k}(z) = H_{n,k}\,P_{n,k}(z) + H_{n,k+1}\,P_{n,k-1}(z), \qquad (2.61)$$

de donde, a su vez, pueden obtenerse fórmulas análogas a las dadas en (2.56) y (2.58) utilizando los polinomios normalizados (2.54) (ref 8). Pero lo verdaderamente interesante es que mediante estas nuevas fórmulas de recursión se llega a las relaciones (ref 8)

$$q_{nk} + e_{nk} = q_{n+1,k} + e_{n,k-1}\ , \qquad (2.62)$$

$$q_{nk} \cdot e_{nk} = q_{n,k+1} \cdot e_{n-1,k}\ , \qquad (2.63)$$

donde e_{nk} y q_{nk} son las mismas que en (2.57) y (2.59), haciendo uso exclusivamente de la siguiente propiedad del polinomio normalizado $p_{n,k}(z)$:

$$p_{n,k}(0) = 1\ . \qquad (2.64)$$

De nuevo conviene observar que las relaciones dadas aquí no son más que las que se motivaron en la primera sección de este capítulo (i.e., (2.31) y (2.32)).

Además, con los polinomios normalizados dados en (2.54) se puede

construir la tabla de Padé, a ellos asociada, para la función F(z). Esta tabla es exactamente igual a la Tabla 2.3, sólo que ahora el elemento (n, k) de la misma nos vendrá dado por el cociente $n_{n,k}(z)/k_{n,k}(z)$ (ver Tabla 2.4). Y como las fórmulas (2.56) y (2.58) valen tanto para $n_{n,k}(z)$ como para $k_{n,k}(z)$, lo que en realidad se ha encontrado son relaciones entre numeradores o denominadores de elementos contiguos de la tabla de Padé; por ello, de ahora en adelante utilizaremos una $p_{n,k}(z)$ para designar o bien a cualquiera de los polinomios $n_{n,k}(z)$,

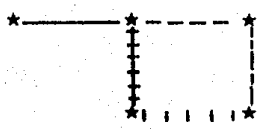
n \ k	1	2	3		— —	— —		k	— —
0	$\frac{1}{k_{01}(z)}$	$\frac{1}{k_{02}(z)}$	$\frac{1}{k_{03}(z)}$		— —	— —		$\frac{1}{k_{0k}(z)}$	— —
1	$\frac{n_{11}(z)}{k_{11}(z)}$	$\frac{n_{12}(z)}{k_{12}(z)}$	$\frac{n_{13}(z)}{k_{13}(z)}$		— —	— —		$\frac{n_{1k}(z)}{k_{1k}(z)}$	— —
2	$\frac{n_{21}(z)}{k_{21}(z)}$	$\frac{n_{22}(z)}{k_{22}(z)}$	$\frac{n_{23}(z)}{k_{23}(z)}$		— —	— —		$\frac{n_{2k}(z)}{k_{2k}(z)}$	— —
⋮	⋮	⋮	⋮					⋮	
⋮	⋮	⋮	⋮					⋮	
n	$\frac{n_{n1}(z)}{k_{n1}(z)}$	$\frac{n_{n2}(z)}{k_{n2}(z)}$	$\frac{n_{n3}(z)}{k_{n3}(z)}$		— —	— —		$\frac{n_{nk}(z)}{k_{nk}(z)}$	— —
⋮	⋮	⋮	⋮					⋮	
⋮	⋮	⋮	⋮					⋮	

TABLA 2.4. TABLA DE PADE

$k_{n,k}(z)$, o bien un elemento arbitrario de la tabla (análoga a la 2.4) que con cualquiera de ellos se construya. Como se verá, la fórmula (2.56) nos relaciona a los siguientes tres elementos de la mencionada tabla

en tanto que (2.58) hace lo propio pero con los elementos

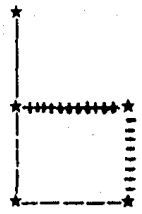
de tal manera que utilizando la primera fórmula dos veces y una la segunda, obtendremos tres ecuaciones que nos relacionan a los siguientes elementos



y de las cuales es factible eliminar a los dos de abajo para obtener la siguiente relación entre los tres de arriba

$$p_{n,k+1}(z) = [1 + (e_{nk} - q_{n,k+1})z]\, p_{n,k}(z) - e_{nk} z\, p_{n,k-1}(z) \ , \qquad (2.65)$$

habiendo utilizado la fórmula (2.62) en el último paso. De la misma manera, una aplicación de (2.56) y dos de (2.58), nos llevan a relacionar a los siguientes elementos



pudiéndose eliminar a los dos de la derecha para obtener la siguiente relación entre tres elementos de una misma columna

$$p_{n+1,k}(z) = [1 + (q_{n,k+1} - e_{nk})\ z\ ]\ p_{n,k}(z) - q_{n,k+1}\ z\ p_{n-1,k}(z), \qquad (2.66)$$

habiéndose utilizado previamente la iguladad (2.63) del algoritmo Q–D.

Observación. De esta última relación se infiere que

$$e_{nk} \xrightarrow[n\to\infty]{} 0 \ , \qquad (2.67)$$

pues de otra manera tendríamos que, para n grande y para $p_{n,k}(z) = k_{n,k}(z)$,

$$k_{n+1,k}(z) \approx (1 - e_{nk}\ z)\ k_{n,k}(z) \ ,$$

es decir, que un polinomio de grado k se aproxima a uno de grado ¡k + 1! En cambio, (2.67) se satisface si $|r_1| \leq |r_2| \leq \ldots \leq |r_k| < |r_{k+1}|$, pues entonces los polinomios $k_{n,k}(z)$ tienen límite para k fija, es decir,

$$k_{n+1,k}(z) \approx k_{n,k}(z) \ ,$$

para n suficientemente grande.

Ahora bien, aplicando (2.67) en la relación (2.65) tendríamos que para n grande y nuevamente para $p_{n,k}(z) = k_{n,k}(z)$

$$k_{n,k+1}(z) \approx (1 - q_{n,k+1}\, z)\, k_{n,k}(z) ,$$

esto es, que

$$q_{n,k+1} \xrightarrow[n\to\infty]{} r_{k+1}^{-1} , \qquad (2.68)$$

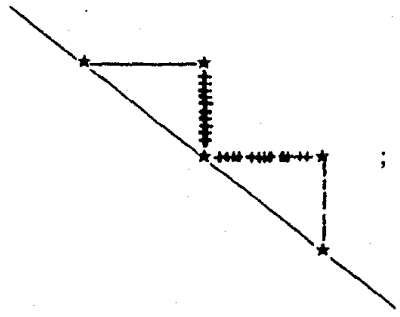
si

$$|r_1| \leq |r_2| \leq \ldots \leq |r_k| < |r_{k+1}| < |r_{k+2}| ,$$

que es precisamente como seleccionamos a la q_{nk} en la primera sección de este capítulo (refs 4, 7).

Por otra parte, la relación entre Teorema de Koenig y Tabla de Padé (Tabla 2.4) se nos presenta ahora más clara que nunca. Aquél nos dice esencialmente que si aproximamos la función meromorfa F(z) mediante la función racional $A_{n,k}(z)/B_{n,k}(z) = N_{n,k}(z)/K_{n,k}(z)$, entonces $K_{n,k}(z)$ tenderá a un polinomio de grado k cuyos ceros son los polos de F(z), cuando $n \to \infty$. Esto, trasladado a la tabla de Padé 2.4 , nos quiere decir que los denominadores de las fracciones de la columna k de dicha tabla tenderán al polinomio de grado k arriba mencionado cuando $n \to \infty$. Algo similar podemos señalar para los renglones de esta tabla: si F(z) tiene n raíces, los numeradores de las fracciones del renglón n tenderán a un polinomio de grado n cuyos ceros son las raíces de F(z), cuando $k \to \infty$; esto se puede apreciar también considerando a la función 1/F(z). La razón de considerar a la tabla 2.4 en vez de la 2.3 es que como en aquélla los polinomios se encuentran ya normalizados, el límite, tanto del numerador como del denominador, será único.

Otra expresión más se puede obtener si aplicamos dos veces la fórmula (2.56) y una la (2.58) de tal forma de abarcar a los siguientes elementos



;

de éstos, se elimina a los dos que no están sobre la diagonal, quedándonos la requerida relación, después de haber aplicado, de nuevo, la fórmula (2.63):

$$p_{n+1,k+1}(z) = [1 - (e_{nk} + q_{n+1,k+1})z]\, p_{n,k}(z) - q_{nk}\, e_{nk}\, z^2\, p_{n-1,k-1}(z). \qquad (2.69)$$

Recalcamos que las relaciones (2.65), (2.66) y (2.69) se satisfacen tanto para numeradores como para denominadores de la tabla de Padé, siendo la más interesante la última, la cual nos permite un desarrollo en fracciones continuas a lo largo de una diagonal de la tabla. Y es aquí donde radica precisamente la importancia de estas fórmulas, ya que, como mencionábamos en la primera sección, es de la expansión en fracciones continuas de la función F(z) de donde surgen de manera natural las cantidades e_{nk}, q_{nk} que determinan al algoritmo Q–D (refs 7, 15).

2.3 DESCRIPCION Y EFICIENCIA DEL ALGORITMO

2.3.1 Descripción del Algoritmo. Hemos visto en las dos primeras secciones de este capítulo otras tantas formas de derivar un mismo método para calcular los polos de una función F(z) meromorfa en $|z| < R$. En virtud de como se definieron ahí las

cantidades e_{nk} y q_{nk}, señalamos la necesidad de que los determinantes de Haenkel no se anularan para garantizar la existencia del modelo. A este respecto cabe señalar que para una función trascendente el esquema Q–D siempre existe, en tanto que para la función racional $F(z) = Q(z)/P(z)$, con Q y P polinomios de grados M y N, respectivamente, únicamente las columnas $q_{n1}, e_{n1}, \ldots, q_{nN}, e_{nN}$ siempre existen para $n \geq$ máx $(0, M - N + 1)$, con $e_{nN} = 0$ (ref 4). Además, para la construcción de la tabla 2.2 hemos supuesto

$$e_{n0} = 0, \quad n \geq 0 ,$$

$$q_{0k} = 0, \quad k > 1 .$$

Veamos cómo podemos aprovechar todo esto para calcular los ceros del polinomio de grado N con coeficientes complejos

$$P(z) = a_0 z^N + a_1 z^{N-1} + \ldots + a_{N-1} z + a_N .$$

Denotemos por r_i, $1 \leq i \leq N$, a las raíces de este polinomio y supongamos que

$$|r_1| \leq |r_2| \leq \ldots \leq |r_N| .$$

Evidentemente podríamos calcular estas raíces calculando los polos de la función $f(z) = 1/P(z)$. Pero como las cantidades q_{nk} convergen en este caso a los recíprocos de tales cantidades, sería mejor considerar de una buena vez a la función cuyos polos sean los recíprocos de los ceros de P(z), esto es, $f(z) = 1/z^N P(z^{-1})$, i.e.,

$$f(z) = \frac{1}{a_0 + a_1 z + \ldots + a_{N-1} z^{N-1} + a_N z^N} .$$

Por lo dicho al principio, el esquema Q–D para esta función consistirá de N columnas q limitadas a derecha e izquierda por sendas columnas e que son idénticamente nulas.

En este caso existe una forma alternativa de calcular el modelo considerando, en vez de la función f(z), a la función

$$F(z) = \frac{1}{f(z)} = a_0 + a_1 z + \ldots + a_{N-1} z^{N-1} + a_N z^N .$$

Entonces la tabla correspondiente puede calcularse por renglones – que como ya señalábamos resulta ser un procedimiento mucho más estable que aquél por columnas – de la siguiente manera (refs 4,14).

	$-\frac{a_1}{a_0}$	0	0	---	0	
0	$\frac{a_2}{a_1}$	$\frac{a_3}{a_2}$	$\frac{a_4}{a_3}$	---	$\frac{a_N}{a_{N-1}}$	0
	q_{11}	q_{12}	q_{13}	---	q_{1N}	
0	e_{11}	e_{12}	e_{13}	---	$e_{1,N-1}$	0
	q_{21}	q_{22}	q_{23}	---	q_{2N}	
0	e_{21}	e_{22}	e_{23}	---	$e_{2,N-1}$	0
⋮	⋮	⋮	⋮		⋮	⋮

TABLA 2.5. MODELO COCIENTE–DIFERENCIA (Q–D)

Como mencionábamos en anterior observación, tendremos que

i) para toda k tal que $|r_k| < |r_{k+1}|$,

$$\lim_{n\to\infty} e_{nk} = 0 ,$$

ii) para toda k tal que $|r_{k-1}| < |r_k| < |r_{k+1}|$,

$$\lim_{n\to\infty} q_{nk} = r_k$$

(refs 4, 7). (Notar que, por lo dicho en el párrafo precedente, las q_{nk} convergen ahora directamente a las raíces del polinomio P).

Surge ahora una pregunta natural: ¿Es éste método tan de escasa generalidad que no se aplica más que a polinomios cuyas raíces satisfacen $|r_1| < |r_2| < \ldots < |r_N|$? La respuesta es negativa ya que para el caso $|r_{k-1}| < |r_k| \leqslant |r_{k+1}| < |r_{k+2}|$, por ejemplo, se tiene que (ref 3, 4)

iii) $$\lim_{n\to\infty} q_{nk}\, q_{n,k+1} = r_k\, r_{k+1} ,$$

$$\lim_{n\to\infty} [q_{n+1,k} + q_{n,k+1}] = r_k + r_{k+1} .$$

Es decir, que para el caso de un par de raíces de módulos posiblemente iguales (un caso muy frecuente y que abarca una situación muy interesante para nosotros: pares de raíces complejas conjugadas), éstas pueden calcularse de columnas contiguas en la tabla Q–D mediante las relaciones anteriores. Pero no sólo esto sino que para módulos de una multiplicidad aun mayor se pueden obtener relaciones similares, aunque más complicadas, a las dadas arriba (ref 4). Aquí nos concretaremos a manejar un programa que nos determina las raíces de un polinomio de grado

mayor o igual a 3 y que se presenten en conjuntos de no más de dos ceros del mismo módulo. Este programa está diseñado para resolver únicamente polinomios reales.

Resumamos el procedimiento que sigue tal programa y cuyo listado damos al final de este capítulo (ref 4).

Si el modelo Q–D no existe para nuestro polinomio original (debido a que algún coeficiente del mismo es cero), el programa introduce la sencilla translación

$$z^* = z - c ,$$

donde c es un número aleatorio generado por la subrutina GENERA que se incluye en dicho programa. Se aplica así el algoritmo al polinomio $P^*(z^*) = P(z^* + c)$, ya que puede probarse que si P tiene algún coeficiente cero, entonces todos los coeficientes de P^* son distintos de cero para valores suficientemente pequeños de $c \neq 0$. Al final, se calculan los ceros r_k del polinomio original a partir de los ceros r_k^* del polinomio P^* mediante la fórmula

$$r_k = r_k^* + c .$$

Hemos dicho que la tabla Q–D se genera por renglones; aún así, sin embargo, el método resulta un tanto inestable, además de que su convergencia es apenas lineal (refs 4, 14). Es por ello que cuando dicha tabla ha quedado dividida en subtablas (en virtud de la condición i) de arriba) que contienen conjuntos de raíces de P de no más de dos de ellas, utilizamos métodos como el de Newton, para las raíces simples, o como el de Bairstow para los pares de raíces del mismo módulo; métodos poderosos de convergencia cuadrática que determinan a las racíces muy aproximadamente y que tan sólo requieren de "buenas" primeras aproximaciones, como aquél-

llas que, simultáneamente a todos los ceros y utilizando como únicos datos a los coeficientes del polinomio, les ofrece el algoritmo Q–D. No discutiremos aquí esos métodos (ref 3). El programa incluye sendas subrutinas para la aplicación de los mismos.

Pueden suceder tres cosas. Primero, que el procedimiento falle desde un principio, es decir, que el modelo Q–D no presente ninguna convergencia, en cuyo caso se dice que el método PRINCIPAL no convergió. Segundo, que alguno de los métodos secundarios falle para la aproximación proporcionada, consignándose así en los resultados obtenidos al final. Finalmente, puede suceder que "todas" las raíces sean calculadas; si así es, se llama a una subrutina (POLORI) que reconstruye el polinomio original a partir de las raíces halladas y como una prueba de la eficacia del método conjunto. Pudo pasar que alguna raíz haya sido calculada más de una vez, en cuyo caso el polinomio será mal reconstruido.

Para finalizar esta parte tan sólo diremos que para satisfacer uno de los requisitos que generalmente se recomiendan para un buen método que calcule los ceros de un polinomio, este programa fue elaborado en su totalidad en doble precisión.

2.3.2 Eficiencia del Programa. El esqueleto del programa en FORTRAN que incluímos al final de este capítulo fue tomado de un artículo de Henrici (ref 4) y adaptado al sistema BORROUGHS 6700 del CIMASS. Las generalidades y las ventajas de este programa quedaron ya expuestas en el párrafo anterior; aquí nos concretaremos a señalar algunos de sus resultados prácticos.

La relativa poca generalidad del algoritmo no nos permitió probar una gran diversidad de polinomios. Sin embargo, la gran cantidad de polinomios aleatorios reales resueltos por el programa (utilizando para generarlos la misma sub-

rutina GENERA mencionada con anterioridad), nos llevaron a determinar los datos óptimos que requiere el mismo (además de los coeficientes) para resolver un determinado polinomio. Estos datos son

EPS: = tolerancia de las primeras aproximaciones calculadas mediante el algoritmo Q–D,

MMAX: = número máximo de renglones que se calculan en el modelo Q–D,

ETA: = tolerancia de las raíces calculadas por el método de Newton o el de Bairstow,

ITMAX: = número máximo de iteraciones permitidas para dichos métodos;

y las cantidades óptimas encontradas

$$\begin{aligned} EPS &= 0.1 \\ MMAX &= 500 \\ ETA &= 0.000\,001 \\ ITMAX &= 30 \ . \end{aligned}$$

Dentro de los polinomios especialmente construidos por nosotros y resueltos por este método se encuentran los siguientes dos. Ambos fueron resueltos con el programa dado por Henrici (ref 4), es decir, en precisión simple, sin reconstruir el polinomio, sin datos óptimos y cuidando de que ninguno tuviese algún coeficiente cero. No obstante, los resultados fueron excelentes y los tiempos de ejecución mínimos, lo que, después de todo, nos permite considerar al algoritmo Q–D como un método poderoso.

1) Polinomio real de grado 12 que tiene por raíces a 7, 6, 5, 4, 3, $1 \pm 2i$, 2, $1 \pm i$, $\pm i$, es decir, seis raíces reales de multiplicidad uno y tres pares de raíces complejas conjugadas.

$$P(z) = z^{12} - 31\,z^{11} + 415\,z^{10} - 3\,187\,z^{9} + 15\,811\,z^{8} +$$
$$- 54\,315\,z^{7} + 134\,953\,z^{6} - 247\,733\,z^{5} +$$
$$+ 338\,428\,z^{4} - 347\,414\,z^{3} + 269\,272\,z^{2} +$$
$$- 150\,840\,z + 50\,400 .$$

Se obtuvo una aproximación mínima de 6 cifras con los siguientes datos: EPS = 0.5, MMAX = 20, ETA = 0.000 01, ITMAX = 10.

2) Polinomio real de grado 12 con tres pares de raíces complejas conjugadas ($1 \pm 2i$, $1 \pm i$, $\pm i$) y tres pares de raíces reales de la misma magnitud (2, 2, 3, 3, 4, 4).

$$P(z) = z^{12} - 22\,z^{11} + 217\,z^{10} - 12\,82\,z^{9} + 5\,113\,z^{8} +$$
$$- 14\,658\,z^{7} + 31\,207\,z^{6} - 50\,102\,z^{5} +$$
$$+ 61\,198\,z^{4} - 57\,248\,z^{3} + 40\,648\,z^{2} +$$
$$- 20\,544\,z + 5\,760 .$$

Los resultados fueron diez raíces calculadas con una aproximación mínima de cuatro cifras y dos en las que el proceso iterativo no convergió. Los datos fueron los siguientes: EPS = 1.0, MMAX = 30, ETA = 0.000 01, ITMAX = 30.

Sin embargo, una vez modificado el programa y con los datos óptimos en la mano, nos dedicamos a probarlo exhaustivamente con polinomios aleatorios reales de grados 3 al 10.

De los muchos programas corridos con el juego de datos óptimos, escogimos aquél que parece darnos resultados representativos. De los 80 poli-

nomios que generó (10 para cada grado del 3 al 10), resolvió 64 (es decir, ¡un 80 por ciento!), dejó sin terminar 5 y en 11 falló la convergencia del modelo Q–D. Estos resultados se aprecian más claramente en la siguiente tabla, en la que se incluyen

n: = grado del polinomio,
NR: = número de polinomios resueltos,
NM: = número de polinomios dejados sin terminar,
NNR: = número de polinomios en los que falló la convergencia del modelo Q–D,
TEJT: = tiempo de ejecución total para polinomios resueltos,
TEJP: = tiempo de ejecución promedio por polinomio resuelto,
APROX: = aproximación estimada a partir de los polinomios reconstruidos.

n	NR	NM	NNR	TEJT (segs)	TEJP (segs)	APROX
3	10	0	0	1.412	0.1412	9 cifras
4	9	1	0	1.707	0.1896	6 cifras
5	10	0	0	3.629	0.3629	5 cifras
6	9	0	1	2.927	0.3252	5 cifras
7	7	1	2	2.853	0.4076	5 cifras
8	6	2	2	4.886	0.8143	5 cifras
9	6	0	4	4.101	0.6835	4 cifras
10	7	1	2	5.101	0.7287	4 cifras
Total	64	5	11	26.616	3.6530	
%	80	6.25	13.75			

TABLA 2.6. TABLA DE POLINOMIOS ALEATORIOS RESUELTOS POR EL ALGORITMO COCIENTE–DIFERENCIA (Q–D)

Con los tiempos dados en la sexta columna de esta tabla elaboramos la gráfica que aparece en la siguiente página y que nos permite apreciar mejor

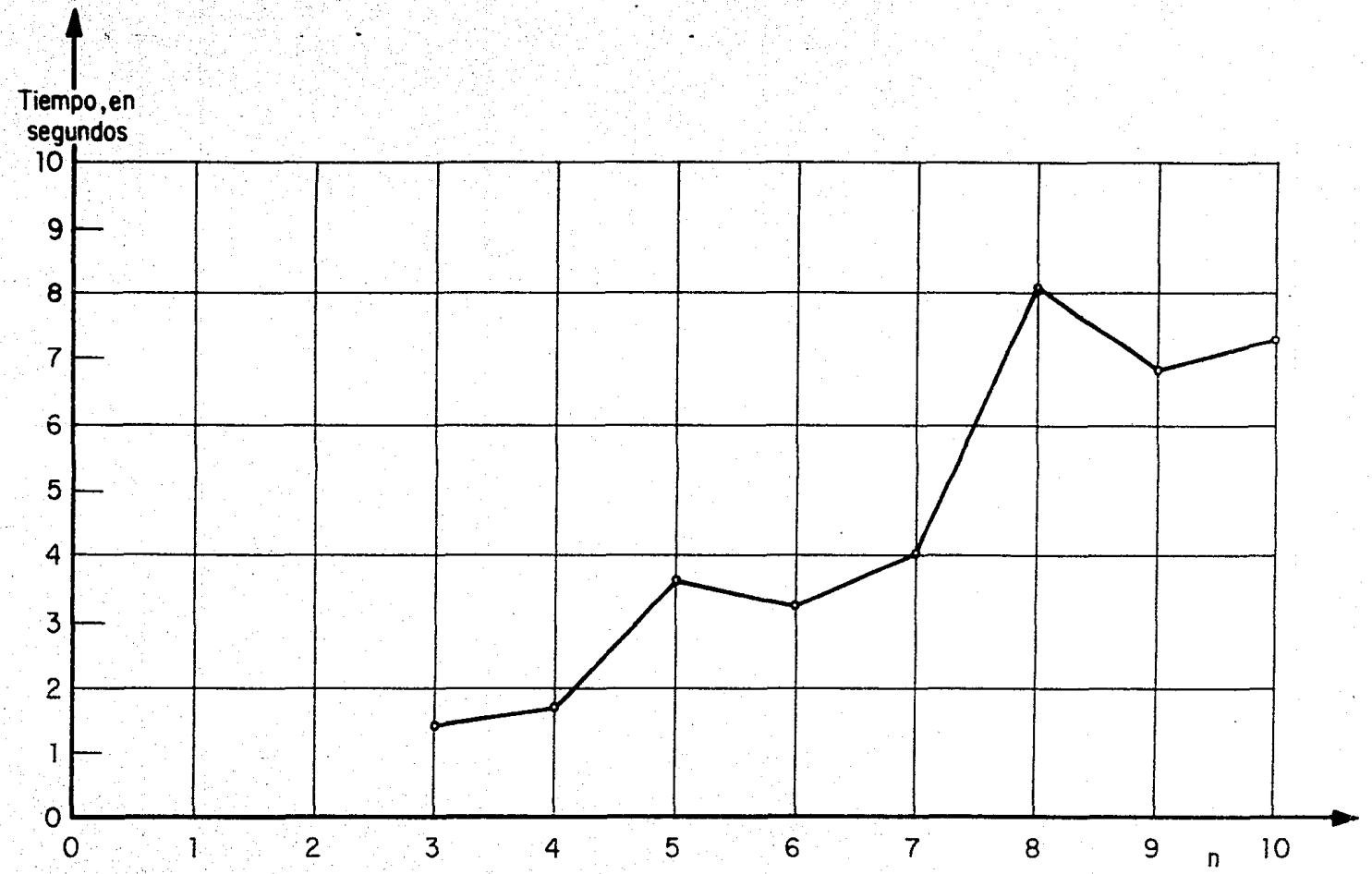


Fig 2.1

el tiempo promedio que tarda el programa para resolver un polinomio de grado n ($3 \leqslant n \leqslant 10$) y compararlo con el que tarda para resolver polinomios de otros grados. Como se observará, los polinomios de grado 8 son especialmente difíciles de resolver; esto lo comprobamos a lo largo de todos nuestros programas.

Para finalizar diremos que en un momento dado llegamos a probar polinomios aleatorios de grados 11 al 20 y tan sólo incrementando el número máximo de renglones que se calculan para el modelo Q–D (MMAX) con respecto a los datos óptimos que dimos con anterioridad. Con MMAX de 1 000 o 1 500 se llegaron a resolver hasta 50 polinomios de un total de 100 (10 para cada grado del 11 al 20).

En las hojas que siguen incluimos el listado del programa discutido a lo largo de esta tercera sección.

2.3.3 Programa en FORTRAN IV para Polinomios Reales

```
                                                    COMPILACION EN FORTRAN G0F00
                                                                  NIVEL II,02,04
                                        FEBRERO     JUEVES/01/73  12:21 PM
                                                    FIB IS 0007 LONG
                                                    START OF SEGMENT 002
FILE  5= LECTOR, UNIT = READER                                      C  002:0000:0
C     PROGRAMA PARA CALCULAR LOS CEROS DE UN POLINOMIO              C  002:0000:0
C                                                                   C  002:0000:0
      DIMENSION IND(49)                                             C  002:0000:0
      DOUBLE PRECISION A(49),Q(49),E(49),QPRIME(49), R(100,100), C1,C(49
     *),ZERO(49),ZERNI(49),XR(100),XI(100),YR(100),Z,S,R            C  002:0000:0
      COMMON ITMAX, KLOOP, P, ETA                                   C  002:0000:0
10000 READ(5,1000,END=62)N                                          C  002:0000:0
      KRA = 0                                                       C  002:000C:0
      KRU = 0                                                       C  002:000C:4
      NP1 = N + 1                                                   C  002:000D:2
      READ (5,1001) (A(I), I = 1, NP1)                              C  002:000E:4
      EPS = 0.10                                                    C  002:001E:2
      NMAX =  500                                                   C  002:0020:3
      ETA = 0.000001                                                C  002:0021:3
      ITMAX = 30                                                    C  002:0024:1
      PRINT 2000, N, (A(I), I = 1, NP1)                             C  002:0025:2
                                                                        FIB IS 0005 LONG
      PRINT 2001, EPS, NMAX, ETA, ITMAX                             C  002:003E:5
      DO 70 I = 1,NP1                                               C  002:0049:3
      IF(A(I)) 70,71,70                                             C  002:004B:0
   70 CONTINUE                                                      C  002:004C:4
      KR = 0                                                        C  002:004E:5
      GO TO 72                                                      C  002:004F:3
   71 KR = 1                                                        C  002:0050:0
      CALL GENERA(2,4,4,2,B,1,9,1,BIEN)                             C  002:0050:4
      C1 = B(1,1)                                                   C  002:0054:3
      C(1) = A(1)                                                   C  002:0055:4
      DO 75 J = 1,N                                                 C  002:0057:1
      DO 73 I = 2,NP1+1-J                                           C  002:0058:0
      C(I) = A(I) + C1*C(I-1)                                       C  002:0059:0
      A(I) = C(I)                                                   C  002:005D:1
   73 CONTINUE                                                      C  002:005F:2
   75 CONTINUE                                                      C  002:0062:2
      PRINT 2002, N, (C(I), I=1,NP1)                                C  002:0064:3
      DO 74 I = 1,NP1                                               C  002:0076:5
      A(I) = C(I)                                                   C  002:0078:0
   74 CONTINUE                                                      C  002:007A:1
C     INICIALIZACION                                                C  002:007C:2
C                                                                   C  002:007C:2
   72 Q(1) = -A(2)/A(1)                                             C  002:007C:2
      DO 100 K = 2, N                                               C  002:007F:0
  100 Q(K) = 0.                                                     C  002:0080:0
      E(1) = 0.                                                     C  002:0083:5
      NM1 = N - 1                                                   C  002:0085:0
      DO 200 K = 1, NM1                                             C  002:0086:2
  200 E(K+1) = A(K+2)/A(K+1)                                        C  002:0087:0
      E(N+1) = 0.                                                   C  002:008C:5
      DO 300 K = 1, N                                               C  002:008E:1
  300 IND(K) = 0.                                                   C  002:008F:0
C     ITERACION                                                     C  002:0092:5
C                                                                   C  002:0092:5
      NMAX = NMAX + 1                                               C  002:0092:5
      N = 0                                                         C  002:0094:1
```

```
 9999 N = N + 1                                                          C  002:0096:1
      DO 410 K = 1, N                                                    C  002:0096:1
  410 QPRIME(K) = E(K+1) + Q(K) - E(K)                                   C  002:0097:0
      DO 420 L = 1, NM1                                                  C  002:009E:0
      E(L+1)=(QPRIME(L+1)*E(L+1))/QPRIME(L)                              C  002:009F:0
      IF(ABS(E(L+1)).GT.EPS) GO TO 10                                    C  002:00A2:4
      IND(L) = IND(L) + 1                                                C  002:00A4:5
      GO TO 420                                                          C  002:00A7:0
   10 IND(L) = 0.                                                        C  002:00A7:3
  420 CONTINUE                                                           C  002:00A9:1
      KLOOP = 1                                                          C  002:00AB:2
   50 IF(IND(KLOOP).GE.5) GO TO 20                                       C  002:00AC:2
      IF(IND(KLOOP+1).GE.5) GO TO 21                                     C  002:00AE:5
      IF(N.NE.NMAX) GO TO 431                                            C  002:00B1:0
      PRINT 2007                                                         C  002:00B2:2
      TEJ = TIME(2)/60.0                                                 C  002:00B6:0
      TES = TIME(3)/60.0                                                 C  002:00B8:0
      PRINT 222, TEJ,TES                                                 C  002:00BA:0
      GO TO 10000                                                        C  002:00C8:0
C     ***************************************                            C  002:00C8:3
  431 DO 430 I = 1, N                                                    C  002:00C8:3
  430 Q(I) = QPRIME(I)                                                   C  002:00CA:0
      GO TO 9999                                                         C  002:00CE:4
   20 KLOOP = KLOOP + 1                                                  C  002:00CF:1
      GO TO 30                                                           C  002:00D0:5
   21 KLOOP = KLOOP + 2                                                  C  002:00D1:2
      IND(N) = 5                                                         C  002:00D3:1
   30 IF((KLOOP+1).LT.N) GO TO 50                                        C  002:00D4:5
C     DECISION SOBRE LA TECNICA QUE SERA USADA                           C  002:00D6:5
C                                                                        C  002:00D6:5
      KLOOP = 1                                                          C  002:00D6:5
   79 IF(IND(KLOOP).LT.5) GO TO 60                                       C  002:00D7:5
      Z = QPRIME(KLOOP)                                                  C  002:00DA:2
      CALL NEWT(N,Z,KR,C1,ZERUR,ZERUI,KRA)                               C  002:00DC:2
      KLOOP = KLOOP + 1                                                  C  002:00E0:5
      GO TO 61                                                           C  002:00E2:3
   60 S = QPRIME(KLOOP) + QPRIME(KLOOP+1)                                C  002:00E3:0
      R = -Q(KLOOP)*QPRIME(KLOOP+1)                                      C  002:00E6:2
      CALL BAIRS(N,R,S,KR,C1,ZERUR,ZERUI,KRU)                            C  002:00E9:5
      KLOOP = KLOOP + 2                                                  C  002:00EE:5
   61 IF(KLOOP.LE.N) GO TO 79                                            C  002:00F0:4
      KRL = KRA + KRU                                                    C  002:00F2:2
      IF(KRL.LT.N) GO TO 11113                                           C  002:00F3:5
      CALL POLURI(ZERUR,ZERUI,A(1),0.0D0,XR,XI,YR,N)                     C  002:00F6:5
      PRINT 2003, (XR(I), XI(I), I=1,NP1)                                C  002:00FB:3
11113 TEJ = TIME(2)/60.0                                                 C  002:010E:5
      TES = TIME(3)/60.0                                                 C  002:0110:5
      PRINT 222, TEJ,TES                                                 C  002:0112:5
      GO TO 10000                                                        C  002:0121:0
   62 CALL EXIT                                                          C  002:0121:3
C     FORMATS                                                            C  002:0122:2
C                                                                        C  002:0122:2
 1000 FORMAT (I3)                                                        C  002:0122:2
 1001 FORMAT(8D10.0)                                                     C  002:0122:2
 2000 FORMAT (8H ENTRADA//5X,4HN = ,I3/5X,12HCOEFICIENTES/(1X,1PD26.16)) C  002:0122:2
 2001 FORMAT (5X,6HEPS = ,E16.8,8H NMAX = ,I6,7H ETA = ,E16.8,
     19H ITMAX = ,I3)                                                    C  002:0122:2
 2002 FORMAT(8H ENTRADA//5X,4HN = ,I3/5X,19HNUEVOS COEFICIENTES/
     *(1X,1PD26.16))                                                     C  002:0122:2
 2003 FORMAT(1H0,22HPOLINOMIO RECUPERADO///(1X,2(1PD26.16,4X))///)       C  002:0122:2
```

```
      22  FORMAT(21HTIEMPO DE EJECUCION =,F9.3,5HSEGS.,3X,29HTIEMPO DE ENTRA-      C  002:0122:2
     1DA Y SALIDA = ,F9.3,5HSEGS.)                                                C  002:0122:2
          END                                                                     C  002:0122:2

                                                  REENTRANT FORMAT SEGMENT 003 IS 0003 LONG
                                                NONREENTRANT FORMAT ARRAY IS 0043 LONG

 002:014F:4   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0151:0   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0152:2   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0153:4   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0155:0   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0156:2   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0157:4   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:0159:0   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 002:015C:1   IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT

                                                  SEGMENT 002 IS 015D LONG
```

```
      SUBROUTINE NEWT(N,Z,KR,C1,ZEROR,ZEROI,KRA)                        C 005:0000:0
C     SUBRUTINA UTILIZANDO LA TECNICA DE NEWTON-RAPHSON                 C 005:0000:0
C                                                                       C 005:0000:0
      DOUBLE PRECISION B(49),C(49),A(49),Z,DELZ,C1,ZEROR(49),ZEROI(49)  C 005:0000:0
      COMMON ITMAX, KLOOP, A, ETA                                       C 005:0000:0
      ITMAX = ITMAX + 1                                                 C 005:0000:0
      NP1 = N + 1                                                       C 005:0001:4
      K = KLOOP                                                         C 005:0003:0
      DO 999 IT = 1, ITMAX                                              C 005:0004:1
C     CALCULO DEL ARREGLO B(N)                                          C 005:0005:0
C                                                                       C 005:0005:0
      B(1) = A(1)                                                       C 005:0005:0
      DO 100 I = 2, NP1                                                 C 005:0006:3
  100 B(I) = A(I) + Z*B(I-1)                                            C 005:0008:0
C     CALCULO DEL ARREGLO C(N)                                          C 005:000E:2
C                                                                       C 005:000E:2
      C(1) = B(1)                                                       C 005:000E:2
      DO 200 J = 2, N                                                   C 005:000F:5
  200 C(J) = B(J) + Z*C(J-1)                                            C 005:0011:0
C     CALCULO DE DELTA Z                                                C 005:0017:4
C                                                                       C 005:0017:4
      DELZ = -B(NP1)/C(N)                                               C 005:0017:4
      Z = Z + DELZ                                                      C 005:001A:5
C     PRUEBA DE CONVERGENCIA                                            C 005:001C:1
C                                                                       C 005:001C:1
      IF(ETA.GE.DABS(DELZ)) GO TO 10                                    C 005:001C:1
  999 CONTINUE                                                          C 005:001E:2
      PRINT 2000, K                                                     C 005:0020:5
      RETURN                                                            C 005:002C:3
   10 KRA = KRA + 1                                                     C 005:002D:0
      IF(KR) 70,71,70                                                   C 005:002E:2
   70 Z = Z + C1                                                        C 005:002F:3
   71 ZEROR(K) = Z                                                      C 005:0030:5
      ZEROI(K) = 0.0                                                    C 005:0033:0
      PRINT 2001, K, Z                                                  C 005:0035:1
      RETURN                                                            C 005:0043:0
C     FORMATS                                                           C 005:0043:3
C                                                                       C 005:0043:3
 2000 FORMAT (6HNEWTON/5H K = ,I3,33H EL PROCESO ITERATIVO NO CONVERGE) C 005:0043:3
 2001 FORMAT(6HNEWTON/5H K = ,I3,5H Z = ,1PD26.16)                      C 005:0043:3
      END                                                               C 005:0043:3
                                        NONREENTRANT FORMAT ARRAY IS 0011 LONG
 005:005C:2  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 005:005D:4  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
                                                SEGMENT 005 IS 005E LONG
```

```
      SUBROUTINE BAIRS(N,R,S,KR,C1,ZEROR,ZEROI,KRU)                     C  006:0000:0
C     SUBRUTINA UTILIZANDO EL METODO DE BAIRSTOW                        C  006:0000:0
C                                                                       C  006:0000:0
      DOUBLE PRECISION A(49),B(49),C(49),S,R,DET,DELS,DELR,REZ,DISC,W,  C  006:0000:0
     *Z1,Z2,C1,ZEROR(49),ZEROI(49)                                      C  006:0000:0
      COMMON ITMAX, KLOOP, A, ETA                                       C  006:0000:0
      ITMAX = ITMAX + 1                                                 C  006:0000:0
      NP1 = N + 1                                                       C  006:0001:4
      K = KLOOP                                                         C  006:0003:0
      DO 999 IT = 1, ITMAX                                              C  006:0004:1
C     CALCULO DEL ARREGLO B(N)                                          C  006:0005:0
C                                                                       C  006:0005:0
      B(1) = A(1)                                                       C  006:0005:0
      B(2) = A(2) + S*B(1)                                              C  006:0006:3
      DO 100 I = 3, NP1                                                 C  006:0009:3
  100 B(I) = A(I) + S*B(I-1) + R*B(I-2)                                 C  006:000B:0
C     CALCULO DEL ARREGLO C(N)                                          C  006:0013:2
C                                                                       C  006:0013:2
      C(1) = B(1)                                                       C  006:0013:2
      C(2) = B(2) + S*C(1)                                              C  006:0014:5
      DO 200 J = 3, N                                                   C  006:0017:4
  200 C(J) = B(J) + S*C(J-1) + R*C(J-2)                                 C  006:0019:0
C     DETERMINANTE CALCULADO Y UTILIZADO PARA CALCULAR DELS Y DELR      C  006:0021:4
      DET = C(N-1)**2 - C(N)*C(N-2)                                     C  006:0021:4
      DELS = (B(N+1)*C(N-2)- B(N)*C(N-1))/DET                           C  006:0027:5
      DELR = (B(N)*C(N) - B(N+1)*C(N-1))/DET                            C  006:002E:0
      R = R + DELR                                                      C  006:0034:0
      S = S + DELS                                                      C  006:0035:2
C     PRUEBA DE CONVERGENCIA                                            C  006:0036:4
C                                                                       C  006:0036:4
      IF(ETA.GE.DABS(DELR).AND.ETA.GE.DABS(DELS)) GO TO 10              C  006:0036:4
  999 CONTINUE                                                          C  006:003A:4
      PRINT 2000, K                                                     C  006:003D:1
      RETURN                                                            C  006:0049:0
C     CONVERGENCIA                                                      C  006:0049:3
C                                                                       C  006:0049:3
   10 KRU = KRU + 2                                                     C  006:0049:3
      REZ = .5*S                                                        C  006:004B:0
      DISC = REZ**2 + R                                                 C  006:004E:0
C     PRUEBA DE SIGNO PARA EL DISCRIMINANTE                             C  006:0050:3
C                                                                       C  006:0050:3
      W = DSQRT(DABS(DISC))                                             C  006:0050:3
      IF(DISC.LT.0.) GO TO 20                                           C  006:0052:2
      Z1 = REZ + W                                                      C  006:0053:3
      KP1 = K + 1                                                       C  006:0054:5
      Z2 = REZ - W                                                      C  006:0056:1
      IF(KR) 70,Z1,70                                                   C  006:0057:3
   70 Z1 = Z1 + C1                                                      C  006:0058:4
      Z2 = Z2 + C1                                                      C  006:005A:0
   71 ZEROR(K ) = Z1                                                    C  006:005B:2
      ZEROI(K ) = 0.0                                                   C  006:005D:3
      ZEROR(KP1) = Z2                                                   C  006:005F:4
      ZEROI(KP1) = 0.0                                                  C  006:0061:5
      PRINT 2001, K, Z1, KP1, Z2                                        C  006:0064:0
      RETURN                                                            C  006:0076:0
   20 IF(KR) 72,73,72                                                   C  006:0076:3
   72 REZ = REZ + C1                                                    C  006:0077:4
   73 ZEROR(K) = REZ                                                    C  006:0079:0
      ZEROI(K) = W                                                      C  006:007B:1
```

```
      ZERO(K+1) = REZ                                                    C  006:007F:1
      ZERO(K+1) = -W                                                     C  006:0081:1
      PRINT 2002, K,REZ,W                                                C  006:0091:0
      RETURN                                                             C  006:0091:3
C     FORMATS                                                            C  006:0091:3
C                                                                        C  006:0091:3
 2000 FORMAT(8HBAIRSTOW/5H K = ,I3,33H EL PROCESO ITERATIVO NO CONVERGE) C  006:0091:3
 2001 FORMAT(8HBAIRSTOW/5H K = ,I3,6H Z1 = ,1PD26.16,7H K+1 = ,I3,       C  006:0091:3
     16H Z2 = ,1PD26.16)                                                 C  006:0091:3
 2002 FORMAT(8HBAIRSTOW/5H K = ,I3,7H REZ = ,1PD26.16,7H IMZ = ,1PD26.16 C  006:0091:3
     *)                                                                  C  006:0091:3
      END                                                                C  006:0091:3
                                                  NONREENTRANT FORMAT ARRAY IS 0021 LONG
 00A:00B1:3 IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 006:00B2:5  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
 006:00B4:1 IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
                                                  SEGMENT 006 IS 0085 LONG
```

```
      SUBROUTINE GENERA(SIGNO,ENTERA,DECIMA,OPCION,A,M,N,RANGO,BIEN)                C 009:0000:0
C SIGNO VALE 2 SI SE QUIEREN NUMEROS CON SIGNO                                      C 009:0000:0
C SIGNO VALE 1 SI SE QUIEREN NUMEROS POSITIVOS                                      C 009:0000:0
C OPCION VALE 1 SI SE QUIERE MATRIZ CON MAGNITUD FIJA                               C 009:0000:0
C OPCION VALE 2 SI SE QUIERE MATRIZ CON MAGNITUD VARIABLE                           C 009:0000:0
C OPCION VALE 3 SI SE QUIERE MATRIZ SIMETRICA CON MAGNITUD FIJA                     C 009:0000:0
C OPCION VALE 4 SI SE QUIERE MATRIZ CON MAGNITUD VARIABLE                           C 009:0000:0
C A ES LA MATRIZ QUE SE GENERA                                                      C 009:0000:0
C M Y N SON LAS DIMENSIONES                                                         C 009:0000:0
C RANGO ES EL RANGO DEL ESPACIO RENGLONES                                           C 009:0000:0
C ENTERA-NUMERO DE ENTEROS PARA OPCION 1 Y 3                                        C 009:0000:0
C DECIMA DECIMALES PARA OPCION 1 Y 3                                                C 009:0000:0
C BIEN VALE 1 SI TODO FUE NORMAL                                                    C 009:0000:0
C BIEN VALE 2 SI SALIO POR ERROR                                                    C 009:0000:0
      DOUBLE PRECISION A(100,100)                                                   C 009:0000:0
      INTEGER RANGO                                                                 C 009:0000:0
      BIEN=1                                                                        C 009:0000:0
      IF((M*RANGO).LT.0) GO TO 10180                                                C 009:0000:4
      NUM=10**(ENTERA-1)                                                            C 009:0002:2
      AIDECI=10.**DECIMA                                                            C 009:0005:0
      MULTI=ENTERA+DECIMA                                                           C 009:0007:1
      ANUM=10**MULTI                                                                C 009:0008:4
      IPAR=TIME(1)                                                                  C 009:000B:0
      IF(OPCION.GE.3) GO TO 10090                                                   C 009:000C:3
      DO 10010 J=1,N                                                                C 009:000D:5
      DO 10010 I=1,RANGO                                                            C 009:000F:0
      GO TO(10090,10110),OPCION                                                     C 009:0010:0
10110 ADI=RANDOM(IPAR)                                                              C 009:0015:3
      IDIG=ADIG*1000                                                                C 009:0017:1
      INTERA=IDIG/100                                                               C 009:0018:3
      IDECI=IDIG-INTERA*100                                                         C 009:0019:4
      IDECIM=IDECI/10                                                               C 009:001B:4
      ISIGNO=IDECI-IDECIM*10                                                        C 009:001C:5
      INTERA=MOD(INTERA,4)                                                          C 009:001E:5
      IDECIM=MOD(IDECIM,5)                                                          C 009:0020:2
      ANUM=10**(INTERA+IDECIM)                                                      C 009:0021:5
      AIDECI=10**IDECIM                                                             C 009:0024:4
10090 AZAR=RANDOM(IPAR)                                                             C 009:0027:0
      IZAR=AZAR*ANUM                                                                C 009:0028:4
      IF(IZAR.NE.0) GO TO 10100                                                     C 009:0029:5
      AZAR=AZAR*10000                                                               C 009:002A:4
      IZAR=AZAR*ANUM                                                                C 009:002C:1
      IF(IZAR.NE.0) GO TO 10100                                                     C 009:002D:2
      AZAR=AZAR*10000                                                               C 009:002E:1
      IZAR=AZAR*ANUM                                                                C 009:002F:4
10100 A(I,J)=IZAR                                                                   C 009:0030:5
      A(I,J)=A(I,J)/AIDECI                                                          C 009:0034:2
      GO TO(10010,10130),SIGNO                                                      C 009:0036:1
10130 IF(MOD(ISIGNO**2),EQ.0) A(I,J)=-A(I,J)                                        C 009:0038:3
10010 CONTINUE                                                                      C 009:0042:4
      DO 10200 I=RANGO+1,M                                                          C 009:0047:0
      ACOL=RANDOM(IPAR)                                                             C 009:0049:0
      ICOL=ACOL*(10.**6)                                                            C 009:004A:4
      ICOL1=ICOL/1000                                                               C 009:004C:5
      ICOL2=ICOL-ICOL1*1000                                                         C 009:004E:1
      IREG1=MOD(ICOL1,RANGO)                                                        C 009:0050:2
      IREG2=MOD(ICOL2,RANGO)                                                        C 009:0051:5
      IS=-1                                                                         C 009:0053:2
      IF(MOD(ICOL1+ICOL2,2).EQ.0) IS=1                                              C 009:0058:1
```

```
      DO 10210 J=1,N                                                    C  009:00580
10210 A(I,J)=A(IREG1+1,J)+IS*A(IREG2+1,J)                               C  009:00590
10200 CONTINUE                                                          C  009:00630
      GO TO 10020                                                       C  009:00650
10000 IF(N.NE.N) GO TO 10030                                            C  009:00661
      DO 10040 I=1,N                                                    C  009:00673
      DO 10040 J=1,N                                                    C  009:00690
      GO TO(10140,10150),(OPCION-2)                                     C  009:006A0
10150 ADIG=RANDOM(IPAR)                                                 C  009:00701
      IDIG=ADIG*1000                                                    C  009:00714
      INTERA=IDIG/100                                                   C  009:00731
      IDECI=IDIG-INTERA*100                                             C  009:00741
      IDECIH=IDECI/10                                                   C  009:00761
      ISIGNO=IDECI-IDECIH*10                                            C  009:00772
      INTERA=MOD(INTERA,4)                                              C  009:00792
      IDECIH=MOD(IDECIH,5)                                              C  009:007A5
      ANUM=10**(INTERA+IDECIH)                                          C  009:007C2
      AIDECI=10**IDECIH                                                 C  009:007F1
10140 AZAR=RANDOM(IPAR)                                                 C  009:00813
      IZAR=AZAR*ANUM                                                    C  009:00831
      IF(IZAR.NE.0) GO TO 10300                                         C  009:00842
      AZAR=AZAR*10000                                                   C  009:00851
      IZAR=AZAR*ANUM                                                    C  009:00861
      IF(IZAR.NE.0) GO TO 10300                                         C  009:00875
      AZAR=AZAR*10000                                                   C  009:00881
      IZAR=AZAR*ANUM                                                    C  009:00911
10300 A(I,J)=IZAR                                                       C  009:00982
      A(I,J)=A(I,J)/AIDECI                                              C  009:009E5
      GO TO(10170,10160),SIGNO                                          C  009:00924
10160 IF(MOD(ISIGNO,2).EQ.0) A(I,J)=-A(I,J)                             C  009:00980
10170 A(J,I)=A(I,J)                                                     C  009:00901
10040 CONTINUE                                                          C  009:00A30
      GO TO 10020                                                       C  009:00A72
10180 PRINT 10190,N,RANGO                                               C  009:00A75
10190 FORMAT(10X,2HN=,I5,6HRANGO=,I5)                                   C  009:00B53
      DIEN=2                                                            C  009:00B53
      GO TO 10020                                                       C  009:00B62
10030 PRINT 10060,N,N                                                   C  009:00B61
      DIEN=2                                                            C  009:00C43
10060 FORMAT(10X,2HN=,I5,2HN=,I5)                                       C  009:00C52
10020 RETURN                                                            C  009:00C52
      END                                                               C  009:00C55

                                        NONREENTRANT FORMAT ARRAY IS 0009 LONG

      009,00DF,1  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
      009,00E0,3  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT

                                                  SEGMENT 009 IS 00E1 LONG
```

```
      SUBROUTINE POLORI(ZEROR,ZEROI,OPR,OPI,XR,XI,YR,N)                 C  00C:0000:0
C     PRUEBA DE LA EXACTITUD DE NUESTRA SOLUCION; RECONSTRUCCION DEL PO- C  00C:0000:0
C     LINOMIO ORIGINAL A PARTIR DE LAS RAICES ENCONTRADAS Y DE SU PRIMER C  00C:0000:0
C     COEFICIENTE.                                                      C  00C:0000:0
      DOUBLE PRECISION ZEROR(49),ZEROI(49),OPR,OPI,XR(100),XI(100),     C  00C:0000:0
     *YR(100),XRO,XIO                                                   C  00C:0000:0
      XRO = OPR                                                         C  00C:0000:0
      XIO = OPI                                                         C  00C:0000:5
      N1 = N + 1                                                        C  00C:0001:A
      XR(N1) = -ZEROR(1)                                                C  00C:0003:0
      XI(N1) = -ZEROI(1)                                                C  00C:0005:5
      XR(N) = 1.0                                                       C  00C:0008:4
      XI(N) = 0.0                                                       C  00C:000A:5
      DO 10 I = 2, N-1                                                  C  00C:000D:0
      XR(I) = 0.0                                                       C  00C:000E:0
      XI(I) = 0.0                                                       C  00C:0010:1
   10 CONTINUE                                                          C  00C:0012:2
      DO 30 J = 2,N                                                     C  00C:0014:5
      DO 20 I = 2,N                                                     C  00C:0016:0
      YR(I) = XR(I)                                                     C  00C:0017:0
      XR(I) = -(ZEROR(J)*XR(I) - ZEROI(J)*XI(I)) + XR(I+1)              C  00C:001A:3
      XI(I) = -(ZEROR(J)*XI(I) + ZEROI(J)*YR(I)) + XI(I+1)              C  00C:0023:5
   20 CONTINUE                                                          C  00C:002D:1
      YR(N1) = XR(N1)                                                   C  00C:002F:2
      XR(N1) = -(ZEROR(J)*XR(N1) - ZEROI(J)*XI(N1))                     C  00C:0032:5
      XI(N1) = -(ZEROR(J)*XI(N1) + ZEROI(J)*YR(N1))                     C  00C:003A:4
   30 CONTINUE                                                          C  00C:0042:3
      XR(1) = 1.0                                                       C  00C:0044:4
      XI(1) = 0.0                                                       C  00C:0046:0
      DO 40 I = 1,N1                                                    C  00C:0047:2
      YR(I) = XR(I)                                                     C  00C:0048:0
      XR(I) = XR(I)*XRO - XI(I)*XIO                                     C  00C:004B:3
      XI(I) = YR(I)*XIO + XI(I)*XRO                                     C  00C:0050:3
   40 CONTINUE                                                          C  00C:0055:3
      RETURN                                                            C  00C:0057:A
      END                                                               C  00C:0058:1
                                                                       SEGMENT 00C IS 0060 LONG
```

```
-6.710000000000000D 00000
-5.770000000000000D-00001
-8.031000000000000D 00003
 5.900000000000000D-00001
-6.419999999999999D-00002
 7.500000000000000D 00000
 2.130000000000000D-00001
 3.699999999999999D 00000
-1.815349999999999D 00001
    EPS =   0.10000000E 00 IMAX =    500 ETA =   0.10000000E-05 IMAX =  30
BAIRSTOW
K =   1 REZ = -1.3034495167631540D-00002 IMZ =  3.4702170135166049D 00001
BAIRSTOW
K =   3 REZ =  2.4384827716019420D-00003 IMZ =  4.0450977963667790D-00001
BAIRSTOW
K =   5 REZ = -2.8360733533615622D-00001 IMZ =  1.9851281077793821D-00001
BAIRSTOW
K =   7 REZ =  2.3803073084946652D-00001 IMZ =  1.8896922531203444D-00001

POLINOMIO RECONSTRUIDO

-6.710000000000000D 00000     -0.000000000000000D 00000
-5.769999999999999D-00001      0.000000000000000D 00000
-8.031000000000000D 00003     -2.775190955115407BD-00024
 5.900000000673911D-00001     -1.136718215215271D-00020
-6.419999999999999D-00002     -1.136718215215271D-00020
 7.500000000000000D 00000      7.104488845095441D-00021
 2.130000000596046D-00001     -4.262693307057266D-00021
 3.699999999999999D 00000      8.840611056369351D-00022
-1.815349999999999D 00001     -1.776122211273861D-00022
TIEMPO DE EJECUCION =    1.027SEGS.   TIEMPO DE ENTRADA Y SALIDA =     0.971SEGS.
```

CAPITULO 3.

ITERACION DE TRES PASOS DE DESPLAZAMIENTO VARIABLE PARA CAL-CULAR LOS CEROS DE UN POLINOMIO COMPLEJO

3.0 INTRODUCCION

De todos los métodos que se estudiaron para la elaboración de ste trabajo el que se describe en el presente capítulo resultó ser muy superior a alquier otro, si bien no tiene las numerosas aplicaciones del Algoritmo Cociente–Diferencia y en su paso final utiliza una fórmula que no es más que una iteración de ewton–Raphson para una cierta función racional. Lo que nos lleva a hacer esta rmación son los resultados obtenidos con el programa implementado para este algoritmo (ref 11): resuelve polinomios de cualquier grado, con coeficientes complejos con cualquier distribución de ceros.

La primera sección está dedicada íntegramente a la motivación algoritmo; en la que le sigue se da su presentación formal y se analiza cada uno

de sus pasos; finalmente, en la tercera sección, se incluyen los magníficos resultados obtenidos con su uso exhaustivo, concluyéndose ésta con el listado del programa en FØRTRAN que se utilizó. Al igual que en el capítulo anterior, una serie de resultados importantes tan sólo se enuncian a todo lo largo del capítulo, concluyéndose éste con un apéndice donde se da la demostración de los mismos.

El método es debido a M. A. Jenkins y J. F. Traub (ref 10). Sin embargo, la motivación del algoritmo dada al principio es completamente original y a lo largo de todo el capítulo se incluyen resultados enteramente nuevos.

3.1 MOTIVACION DEL ALGORITMO

En lo que sigue consideraremos al polinomio con coeficientes complejos

$$P(z) = z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n = \prod_{i=1}^{r} (z - \rho_i)^{m_i} \quad , \qquad (3.1)$$

donde $m_1 + m_2 + \ldots + m_r = n$, y trataremos de obtener un método para calcular sus raíces.

Para resolver este problema se nos presentan dos casos:

α) Raíces simples de módulos posiblemente iguales, es decir, $m_i = 1, 1, 2, \ldots, n$ y

$$|\rho_1| \leq |\rho_2| \leq \ldots \leq |\rho_n| \; ;$$

β) Raíces múltiples,* esto es,

$$|\rho_1| \leq |\rho_2| \leq \ldots \leq |\rho_r|, \; r < n \, .$$

* Ver la parte final de este capítulo.

Por ahora consideraremos tan sólo el primer caso dejando el segundo para la última parte de este capítulo. SIN EMBARGO, AL FINAL NOTAREMOS QUE EL ALGORITMO ES EXACTAMENTE EL MISMO PARA AMBOS CASOS.

La idea esencial del algoritmo es la de construir una sucesión de polinomios $\{H(\lambda, s_\lambda, z)\}$-donde λ es un natural y $\{s_\lambda\}$ una sucesión de números complejos- de grado n - 1 y tal que

$$\tilde{H}(\lambda, s_\lambda, z) \longrightarrow \frac{P(z)}{z - \rho_j}, \quad \lambda \to \infty , \qquad (3.2)$$

donde $\tilde{H}(\lambda, s_\lambda, z)$ no es más que el polinomio $H(\lambda, s_\lambda, z)$ dividido por el coeficiente de la potencia mayor de z (ref 10).

Es claro a partir de esta última expresión que si logramos construdicha sucesión tendremos determinada automáticamente una aproximación a la raíz de P(z), pues como P(z) y $\tilde{H}(\lambda, s_\lambda, z)$ son mónicos y de grados n y n - 1, respectivamente, al efectuar el cociente $P(z)/\tilde{H}(\lambda, s_\lambda, z)$, obtendremos una expresión del siguiente tipo

$$\frac{P(z)}{\tilde{H}(\lambda, s_\lambda, z)} = z - a + \frac{R(z)}{\tilde{H}(\lambda, s_\lambda, z)} , \qquad (3.3)$$

de R(z) es un polinomio de grado menor que n - 1. Pero, en virtud de (3.2), remos que para λ suficientemente grande

$$\frac{P(z)}{\tilde{H}(\lambda, s_\lambda, z)} \approx z - a \approx z - \rho_j , \qquad (3.4)$$

manera que

$$\rho_j \approx a \approx z - \frac{P(z)}{\widetilde{H}(\lambda, s_\lambda, z)} \quad . \qquad (3.5)$$

1.1 Construcción de la Sucesión $[H(\lambda, s_\lambda, z)]$. Sea

$$P_i(z) = \frac{P(z)}{P'(\rho_i)\ (z - \rho_i)} , \qquad (3.6)$$

e donde, como

$$P'(z) = \sum_{i=1}^{n} \left(\prod_{k \neq i} (z - \rho_k) \right) , \qquad (3.7)$$

s claro que

$$P_i(\rho_k) = \delta_{ik}, \quad i, k = 1, 2, \ldots, n . \qquad (3.8)$$

Hagamos ahora que $H(\lambda, s_\lambda, z)$ sea una combinación lineal los polinomios $P_i(z)$, es decir,

$$H(\lambda, s_\lambda, z) = \sum_{i=1}^{n} c_i(\lambda, s_\lambda)\ P_i(z) , \qquad (3.9)$$

de los coeficientes $c_i(\lambda, s_\lambda)$ dependerán de los parámetros λ, s_λ y veremos la forapropiada de escogerlos. Claramente los polinomios $H(\lambda, s_\lambda, z)$ son, a lo más, de o $n - 1$, pues las $P_i(z)$ son de grado exactamente $n-1$. Observemos, por otra e, que el coeficiente de z^{n-1} en $H(\lambda, s_\lambda, z)$ nos viene dado por

$$m(\lambda, s_\lambda) = \sum_{i=1}^{n} \frac{c_i(\lambda, s_\lambda)}{P'(\rho_i)} , \qquad (3.10)$$

de tal manera que cuando $H(\lambda, s_\lambda, z)$ sea de grado $n - 1$ y queramos hacerlo mónico tendremos que construirlo de acuerdo a la siguiente relación (posteriormente probaremos que si λ es grande, entonces $m(\lambda, s_\lambda) \neq 0$)

$$\tilde{H}(\lambda, s_\lambda, z) = \frac{H(\lambda, s_\lambda, z)}{m(\lambda, s_\lambda)} = \frac{\sum_{i=1}^{n} c_i(\lambda, s_\lambda) P_i(z)}{\sum_{i=1}^{n} \frac{c_i(\lambda, s_\lambda)}{P'(\rho_i)}} \quad . \qquad (3.11)$$

La conveniencia de que los polinomios de la sucesión $[H(\lambda, s_\lambda, z)]$ sean mónicos, la veremos al responder la siguiente pregunta: ¿Cómo podemos conseguir que $\tilde{H}(\lambda, s_\lambda, z) \to P(z)/(z - \rho_j)$, cuando $\lambda \to \infty$?

De acuerdo con las reglas del Cálculo para los límites en ∞ tenemos que si $c_j(\lambda, s_\lambda) > c_i(\lambda, s_\lambda)$ para toda $i \neq j$ y para toda λ

$$(\lambda, s_\lambda, z) = \frac{\sum_{i=1}^{n} \frac{c_i(\lambda, s_\lambda)}{c_j(\lambda, s_\lambda)} P_i(z)}{\sum_{i=1}^{n} \frac{c_i(\lambda, s_\lambda)}{c_j(\lambda, s_\lambda)} \cdot \frac{1}{P'(\rho_i)}} = \frac{P_j(z) + \sum_{i \neq j} \frac{c_i(\lambda, s_\lambda)}{c_j(\lambda, s_\lambda)} P_i(z)}{\frac{1}{P'(\rho_j)} + \sum_{i \neq j} \frac{c_i(\lambda, s_\lambda)}{c_j(\lambda, s_\lambda)} \cdot \frac{1}{P'(\rho_i)}} \quad ;$$

ADEMAS SE SATISFACE LA IMPORTANTE CONDICION –COMO SE VERA, BASICA EN NUESTROS PROPOSITOS DE CONVERGENCIA–

$$\frac{c_i(\lambda, s_\lambda)}{c_j(\lambda, s_\lambda)} \xrightarrow[\lambda \to \infty]{} 0 \quad , \qquad (3.12)$$

toda $i \neq j$, entonces (ref 10)

$$(\lambda, s_\lambda, z) \xrightarrow[\lambda \to \infty]{} \frac{P_j(z)}{\frac{1}{P'(\rho_j)}} = \frac{P(z) / P'(\rho_j)(z - \rho_j)}{1 / P'(\rho_j)} = \frac{P(z)}{z - \rho_j} \quad .$$

.2 **Selección de los Coeficientes** $c_i(\lambda, s_\lambda)$. Una manera de conseguir que la con-
ón (3.12) se cumpla, es haciendo que

$$\frac{c_i(\lambda, s_\lambda)}{c_j(\lambda, s_\lambda)} = \frac{c_i(\lambda)}{c_j(\lambda)} = \frac{\rho_i^{-\lambda}}{\rho_j^{-\lambda}} = \left(\frac{\rho_i}{\rho_j}\right)^{-\lambda} , \qquad (3.13)$$

$|\rho_j| < |\rho_i|$ para toda $i \neq j$; en particular esto se lograría si ρ_j fuese la raíz
módulo más pequeño, esto es, si $\rho_j = \rho_1$ (la interpretación que sigue nos permi-
manejar el caso con raíces de módulos iguales).

Tomando módulos en (3.13)

$$\frac{|c_i(\lambda)|}{|c_j(\lambda)|} = \left(\frac{|\rho_i|}{|\rho_j|}\right)^{-\lambda} , \qquad (3.14)$$

expresión entre paréntesis –cociente del cual depende la convergencia– se puede
erpretar como la distancia relativa de la raíz ρ_i al origen con respecto a la raíz
¿Qué pasaría si cambiásemos de origen, es decir, si

$$c_i(\lambda, s_\lambda) = c_i(\lambda, s) = (\rho_i - s)^{-\lambda} \; ? \qquad (3.15)$$

omitir el subíndice de s_λ en $c_i(\lambda, s_\lambda)$ y escribir $c_i(\lambda, s)$, queremos dar a entender
el nuevo origen s será el mismo para cualquier λ hasta entonces considerada, i.e.,
$s_1 = s_2 = \ldots = s_\lambda$).

Como estamos suponiendo raíces simples, siempre podremos escoger
tal que (3.15) nos determine un cero ρ_j tal que

$$|\rho_j - s| < |\rho_i - s| \ , \qquad (3.16)$$

ara toda $i \neq j$, de tal manera que

$$\tilde{H}(\lambda, s, z) \xrightarrow[\lambda \to \infty]{} \frac{P(z)}{z - \rho_j} \ .$$

Notemos que este resultado permanece válido si para cada λ se-eccionamos un nuevo origen s_λ, pero de tal forma que se conserve

$$|\rho_j - s_\lambda| < |\rho_i - s_\lambda| \ , \qquad (3.17)$$

ara toda $i \neq j$ y para toda λ. Las $c_i(\lambda, s_\lambda)$ estarían entonces dadas por

$$c_i(\lambda + 1, s_{\lambda+1}) = c_i(\lambda, s_\lambda)(\rho_i - s_{\lambda+1})^{-1}, \qquad (3.18)$$

e donde, si hacemos $c_i(0, s_o) = Q(\rho_i)$, obtendremos

$$c_i(\lambda + 1, s_{\lambda+1}) = Q(\rho_i) \prod_{k=1}^{\lambda+1} (\rho_i - s_k)^{-1} \ , \qquad (3.19)$$

nde $Q(z)$ es un polinomio de grado menor o igual que $n - 1$ que no tiene ceros común con $P(z)$ y cuya utilidad veremos posteriormente cuando queramos calcu-las $H(\lambda, s, z)$. (Dados n valores $Q(\rho_1), Q(\rho_2), \ldots, Q(\rho_n)$, el polinomio $Q(z)$ grado $n - 1$ queda determinado).

3 H(λ, s, z) es un Polinomio de Grado n − 1. Señalamos anteriormente que arfamos que, para λ suficientemente grande, $m(\lambda, s) \neq 0$. Hagámoslo ahora,

bando de paso con ello que H (λ, s, z) es un polinomio de grado n − 1.

Sabemos que

$$m(\lambda, s) = \sum_{i=1}^{n} \frac{c_i(\lambda, s)}{P'(\rho_i)}$$

$$= \sum_{i=1}^{n} \frac{Q(\rho_i)\ (\rho_i - s)^{-\lambda}}{P'(\rho_i)}$$

$$= \frac{Q(\rho_j)}{P'(\rho_j)} (\rho_j - s)^{-\lambda} \left[1 + \sum_{i \neq j} \frac{P'(\rho_j)}{Q(\rho_j)} \frac{Q(\rho_i)}{P'(\rho_i)} \left(\frac{\rho_i - s}{\rho_j - s} \right)^{-\lambda} \right],$$

donde, como

$$|\rho_j - s| < |\rho_i - s|,$$

a toda $i \neq j$, existirá un natural λ_o (N_o) tal que

$$R_j(\lambda, s) = \sum_{i \neq j} \left| \frac{P'(\rho_j)}{Q(\rho_j)} \frac{Q(\rho_i)}{P'(\rho_i)} \right| \left| \frac{\rho_i - s}{\rho_j - s} \right|^{-\lambda} < \frac{1}{N_o}, \quad N_o \geq 2, \qquad (3.20)$$

toda $\lambda \geq \lambda_o$ (N_o) y, por lo tanto,

$$m(\lambda, s) \neq 0, \ \lambda \geq \lambda_o, \qquad (3.21)$$

o cual conseguiremos que los polinomios de la sucesión $[\tilde{H}(\lambda, s, z)]$ sean mó-
(ref 10).

1.4 Construcción de dos Sucesiones que Convergen a una Raíz de P(z). Una primera ɔrma de construir una sucesión que converja a una raíz de P(z) la comenzamos a ɛr al principio de esta primera sección de este capítulo. Dijimos entonces que la ıcesión $[\widetilde{H}(\lambda, s, z)]$ nos determinaba una aproximación de la raíz ρ_j de P(z) de siguiente manera

$$\rho_j \approx a \approx z - \frac{P(z)}{\widetilde{H}(\lambda_o, s, z)}, \qquad (3.22)$$

donde una primera aproximación nos vendría dada por

$$a_1 = s - \frac{P(s)}{\widetilde{H}(\lambda_o, s, s)}, \qquad (3.23)$$

ıesto que s es ya una aproximación de ρ_j) siendo λ_o y s las mismas que mencio-ımos en el párrafo anterior. Evidentemente una mejor aproximación a_2 de ρ_j se ɔtendrá si aplicamos (3.22) en $z = a_1$ y para $\lambda_1 = \lambda_0 + 1$, i.e.,

$$a_2 = a_1 - \frac{P(a_1)}{\widetilde{H}(\lambda_o + 1, s, a_1)}. \qquad (3.24)$$

ntinuando de la misma manera, obtenemos la fórmula de recurrencia

$$a_{k+1} = a_k - \frac{P(a_k)}{\widetilde{H}(\lambda_o + k, s, a_k)}, \qquad (3.25)$$

nos dará la sucesión requerida faltándonos únicamente verificar que está bien de-a, es decir, que

$$\tilde{H}(\lambda_o + k, s, a_k) \neq 0 . \qquad (3.26)$$

Veremos al final de este capítulo que, en efecto, esto se cumple, ndo el argumento para probarlo análogo a aquél que utilizamos para demostrar e $m(\lambda, s) \neq 0$.

Una segunda manera de construir otra sucesión que converja a a raíz de P(z) es mediante el método de Newton. Partamos de nuevo de la roximación inicial a_1 dada por (3.23). Observemos que si a_1 está suficientemente cerca ρ_j, entonces podemos aplicar el método Newton-Raphson a la función $P(z)/\tilde{H}(\lambda_o, s, z)$ enerar una sucesión que converja a ρ_j, pues esta función y P(z) tienen a ρ_j por raíz co-n; más aun, ambas funciones tienen exactamente las mismas raíces. Esta sucesión que-ía de la forma siguiente

$$a_2 = a_1 - \frac{P(a_1)/\tilde{H}(\lambda_o, s, a_1)}{[P(z)/\tilde{H}(\lambda_o, s, z)]'_{z=a_1}} , \qquad (3.27)$$

n general

$$a_{k+1} = a_k - \frac{P(a_k)/\tilde{H}(\lambda_o, s, a_k)}{[P(z)/\tilde{H}(\lambda_o, s, z)]'_{z=a_k}} , \qquad (3.28)$$

$$H(\lambda_0, s, z) = \sum_{i=1}^{n} c_i(\lambda_0, s) P_i(z).$$

Estas serían, pues, dos formas posibles de generar sucesiones que rjan a ρ_j, pero, desgraciadamente, ambos procedimientos ignoran el trabajo he-n el otro; por ejemplo, el segundo procedimiento no toma en cuenta lo hecho

ı la sucesión $[H(\lambda, s, z)]$ en el primero. Así pues, sería conveniente un método
: utilizara, simultáneamente, las magníficas cualidades del método de Newton y las
menos apreciables de la sucesión $[H(\lambda, s, z)]$. Recalquemos antes un hecho
portante dado en la siguiente observación.

Observación (A). La sucesión de funciones racionales $[\frac{P(z)}{\widetilde{H}(\lambda, s, z)}]$,
$\geq \lambda_0$, es tal que $P(z)/\widetilde{H}(\lambda, s, z)$ tiene exactamente los mismos ceros que $P(z)$.

Notemos ahora que al aplicar la fórmula de Newton a la función
$)/\widetilde{H}(\lambda_o, s, z)$ y con la aproximación a_1 obtuvimos la relación (3.27); pero preci-
ente lo dicho en el párrafo anterior y en la observación que le siguió nos sugie-
plicar dicho método a la función $P(z)/\widetilde{H}(\lambda_0 + 1, s, z)$ que resulta una mejor
oximación al factor lineal $z - \rho_j$. Tendríamos entonces que

$$a_2 = a_1 - \frac{P(a_1)/\widetilde{H}(\lambda_o + 1, s, a_1)}{[P(z)/\widetilde{H}(\lambda_0 + 1, s, z)]'_{z=a_1}},$$

sí sucesivamente hasta obtener la fórmula general

$$a_{k+1} = a_k - \frac{P(a_k)/\widetilde{H}(\lambda_o + k, s, a_k)}{[P(z)/\widetilde{H}(\lambda_o + k, s, z)]'_{z=a_k}}, \qquad (3.29)$$

e

$$H(\lambda_0 + k, s, z) = \sum_{i=1}^{n} c_i(\lambda_0 + k, s)\, P_i(z).$$

Esta forma de generar una sucesión que converja a ρ_j es ya bas-
superior a cualquiera de las dos que dimos en un principio; sin embargo, no

todavía la óptima. Se siente como que no estamos utilizando simultáneamente dos procedimientos originales, sino calculando, por un lado, a las $H(\lambda_0 + k, s, z)$, diendo pasar, por otro lado, de una aproximación inicial a_1 a una mejor aproxiación que a_2 con tan sólo utilizar la última $H(\lambda_0 + k, s, z)$ calculada. Uno ensa entonces que la mejor manera de generar la requerida sucesión que converja ρ_j pudiera consistir en un proceso combinado que alternara los pasos de los dos ocedimientos originales, aprovechando el segundo las cualidades del primero y viceersa (ref 10).

Antes de pasar a resolver este problema, hagamos una importante servación.

Observación (B). En la última sucesión que generamos, utilizamos sucesión de polinomios $\{H(\lambda, s, z)\}$, con

$$H(\lambda, s, z) = \sum_{i=1}^{n} c_i(\lambda, s) P_i(z), \qquad (3.30)$$

nde s es un número complejo con la siguiente propiedad

$$|\rho_j - s| < |\rho_i - s|, \qquad (3.31)$$

a toda $i \neq j$. Además, la rapidez con que $\tilde{H}(\lambda, s, z)$ tiende a $P(z)/(z - \rho_j)$ de-de de los cocientes

$$\left| \frac{\rho_j - s}{\rho_i - s} \right|, \quad i \neq j,$$

al forma que entre más pequeños sean éstos, mejor aproximará $\tilde{H}(\lambda, s, z)$ a $/(z - \rho_j)$. En otras palabras, si s y s' son "buenas" aproximaciones a ρ_j y s'

mejor aproximación a ρ_j que s, entonces $\widetilde{H}(\lambda, s', z)$ es mejor aproximación a $z)/(z - \rho_j)$ que $\widetilde{H}(\lambda, s, z)$.

Cabe preguntarse después de esta observación por qué, en vez de lcular las $H(\lambda_0 + k, s, z)$ en forma independiente y con s fija, no se calcularon en rma alternada con las a_{k+1} utilizando, en lugar de s, la a_k recién calculada y enguida calculando a_{k+1} con la $H(\lambda_o + k, a_k, z)$ así obtenida. Siguiendo este proso, obtendríamos

$$a_1 = s - \frac{P(s)/\widetilde{H}(\lambda_o, s, s)}{[P(z)/\widetilde{H}(\lambda_o, s, z)]'_{z=s}} ,$$

$$a_2 = a_1 - \frac{P(a_1)/\widetilde{H}(\lambda_o + 1, a_1, a_1)}{[P(z)/\widetilde{H}(\lambda_o + 1, a_1, z)]'_{z=a_1}} ,$$

en general (ref 10)

$$a_{k+1} = a_k - \frac{P(a_k)/\widetilde{H}(\lambda_o + k, a_k, a_k)}{[P(z)/\widetilde{H}(\lambda_o + k, a_k, z)]'_{z=a_k}} , \qquad (3.32)$$

ıde

$$H(\lambda_o + k, a_k, z) = \sum_{i=1}^{n} c_i(\lambda_o + k, a_k) P_i(z) . \qquad (3.33)$$

Observación (C). La sucesión que hemos construido resulta ser la aceptable desde todo punto de vista, excepto por el hecho de que alguien poobjetarnos que requiere de demasiados cálculos. Sin embargo, nosotros responde- con las siguientes afirmaciones definitivas:

Afirmación (I).

$$\frac{P(a_k)/\widetilde{H}(\lambda_o + k, a_k, a_k)}{[P(z)/\widetilde{H}(\lambda_o + k, a_k, z)]'_{z=a_k}} = \frac{P(a_k)}{\widetilde{H}^*(\lambda_o + k + 1, a_k, a_k)}, \qquad (3.34)$$

ando únicamente por probar que $\widetilde{H}^*(\lambda_o + k + 1, a_k, a_k) \neq 0$.

De esta última relación, el algoritmo se reduce a calcular

$$a_1 = s - \frac{P(s)}{\widetilde{H}(\lambda_o + 1, s, s)},$$

$$a_2 = a_1 - \frac{P(a_1)}{\widetilde{H}(\lambda_o + 2, a_1, a_1)},$$

$$\vdots \qquad\qquad \vdots$$

$$a_{k+1} = a_k - \frac{P(a_k)}{\widetilde{H}(\lambda_o + k + 1, a_k, a_k)},$$

le

$$H(\lambda_o + k + 1, a_k, z) = \sum_{i=1}^{n} c_i(\lambda_o + k + 1, a_k)\, P_i(z).$$

Afirmación (II).

$$_{o} + k+1, a_k, z) = \frac{1}{z - a_k}\left[H(\lambda_o + k, a_{k-1}, z) - \frac{H(\lambda_o + k, a_{k-1}, a_k)}{P(a_k)} P(z)\right]^* \qquad (3.35)$$

10).

id .

Para finalizar esta primera sección del capítulo diremos tan sólo que s querido que en ella quede clara la idea del algoritmo y no su rigurosa presen- analítica, pues consideramos que lo primero es lo más importante. Con esto ieremos decir que la parte analítica del algoritmo resulte secundaria, ni mucho s. Es más, cuando uno trata con la parte esencialmente matemática de dicho tmo es cuando termina de entenderlo completamente. Sin embargo, como es- rte resulta un poco tediosa, decidimos incluirla hasta el final para que no inter- con nuestros propósitos principales (ref 10).

EL ALGORITMO

Hemos motivado en la primera sección un algoritmo para resolver oblema general de calcular los ceros del polinomio complejo

$$P(z) = z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n, \qquad (3.36)$$

íces $\rho_1, \rho_2, \ldots, \rho_r$ de multiplicidades $m_1, m_2, \ldots, m_r$, respectivamente, es que

$$|\rho_1| \leq |\rho_2| \leq \ldots \leq |\rho_r|, r < n. \qquad (3.37)$$

odremos, sin pérdida de generalidad, que 0 no es raíz de P(z) ya que si así fuera nos deflacionar inmediatamente el polinomio hasta obtener uno que no tenga or raíz.

Una vez motivado el algoritmo, procedamos a su presentación il y expliquemos brevemente cada uno de sus pasos (ref 10).

Algoritmo (Iteración de Tres Pasos de Desplazamiento Variable para lar los Ceros de un Polinomio).

Primer Paso (Sin Desplazamiento).

$H(0, 0, z) = P'(z)$ *,

$$H(\lambda + 1, 0, z) = \frac{1}{z - 0}[H(\lambda, 0, z) - \frac{H(\lambda, 0, 0)}{P(0)} P(z)], \qquad (3.38)$$

$0, 1, \ldots, M - 1.$

Este paso tiene como finalidad el ponernos en contacto con los os de módulo más pequeño del polinomio (3.36). Es por ello que para éste no uerimos de ningún desplazamiento y la duración del mismo será corta, tan sólo a obtener una H(M, 0, z) adecuada con la cual iniciar el segundo paso, siendo M úmero de iteraciones llevadas a cabo en este primer paso. La M del programa o en la tercera sección es igual a 5.

Segundo Paso (Con Desplazamiento Fijo).

Calcular una cota inferior $\beta > 0$ de los módulos de las raíces de) y seleccionar una s tal que $|s| = \beta$ y tal que

$$|\rho_j - s| < |\rho_i - s|, \qquad (3.39)$$

a toda $i \neq j$, siendo ρ_j una de las raíces de módulo más pequeño. Enseguida, nando a s como nuevo origen, calcular

$$H(M + 1, s, z) = \frac{1}{z - s}[H(M, 0, z) - \frac{H(M, 0, s)}{P(s)} P(z)],$$

bid.

$$H(\lambda + 1, s, z) = \frac{1}{z - s}\left[H(\lambda, s, z) - \frac{H(\lambda, s, s)}{P(s)} P(z)\right] , \qquad (3.40)$$

$$\lambda = M + 1, M + 2, \ldots, L - 1.$$

Este segundo paso es ya de mayor trascendencia, iniciándose con elección de un nuevo origen s que en el caso particular del paso anterior fue ado como s = 0. Como las H(λ, 0, z) fueron construidas de tal manera de ob- r convergencia hacia las raíces pequeñas de (3.36), la razón principal de seleccio- un nuevo origen es hasta cierto punto obvia y aparentemente la única: conseguir alguna de las raíces pequeñas de (3.36) sea la más cercana a s, resolviendo, de), el serio problema de los módulos cercanos. Sin embargo, ésta no es la única ón y en el tercer paso veremos por qué.

Para que se satisfaga este primer cometido de la s, debemos selec- iarla de acuerdo al siguiente criterio. Hallar una cota inferior $\beta > 0$ de los mó- os de las raíces de (3.36) calculando la única raíz positiva del polinomio

$$z^n + |a_1| z^{n-1} + \ldots + |a_{n-1}| z - |a_n| ,^* \qquad (3.41)$$

el método de Newton-Raphson. Inmediatamente después seleccionar a la s sobre círculo de radio β, i. e., $|s| = \beta$, de tal manera que

$$|\rho_j - s| < |\rho_i - s| , \quad i \neq j . \qquad (3.42)$$

ha selección se realiza de manera aleatoria —utilizando una distribución uniforme—, liéndose demostrar que

$$|\rho_j| \leq 3 |\rho_1| . \qquad (3.43)$$

---

*Ibid.

ta relación nos dice que el proceso satisface uno de los requisitos para resultar :able (ref 17).

Con respecto a este segundo paso tan sólo nos falta seleccionar número de veces L–M en que hemos de aplicarlo. La manera de hacerlo es ndo por terminado este paso cuando se satisfaga la simple y débil prueba de nvergencia

$$\begin{aligned} |t_{\lambda+1} - t_\lambda| &\leq \frac{1}{2}|t_\lambda|, \\ |t_{\lambda+2} - t_{\lambda+1}| &\leq \frac{1}{2}|t_{\lambda+1}|, \end{aligned} \tag{3.44}$$

nde

$$t_\lambda = s - \frac{P(s)}{\tilde{H}(\lambda, s, s)} . \tag{3.45}$$

esta prueba de convergencia no tiene éxito, λ alcanza un límite dado y se selec- ona una nueva s; si nuevamente la convergencia falla, λ alcanza otro límite dado, stinto del anterior, y se selecciona otra s, y así sucesivamente hasta que se alcan- la prueba de convergencia (ref 10).

Tercer Paso (Con Desplazamiento Variable).

Calcular

$$s_L = s - \frac{P(s)}{\tilde{H}(L, s, s)},$$

después, haciendo $s = s_{L-1}$, aplicar alternativamente las fórmulas de recurrencia

$$H(\lambda + 1, s_\lambda, z) = \frac{1}{z - s_\lambda} \left[H(\lambda, s_{\lambda-1} z) - \frac{H(\lambda, s_{\lambda-1}, s_\lambda)}{P(s_\lambda)} P(z) \right], \qquad (3.46)$$

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\tilde{H}(\lambda + 1, s_\lambda, s_\lambda)}, \qquad (3.47)$$

$\lambda = L, L + 1, \ldots$.

Una vez terminado el segundo paso contamos ya con la H(L, s, z) ɔpiada con que iniciaremos este último paso. Lo que hace esta fase final del altmo es aprovechar de una manera aun más eficiente la idea del paso anterior, o utilizar la misma fórmula de recurrencia que para ese paso sólo que ahora toıdo un nuevo origen s_λ en cada iteración. ¿Cómo obtener un nuevo origen s_λ, próximo a la raíz ρ_j que el $s_{\lambda-1}$ anterior, para aplicarlo en el cálculo de + 1, s_λ, z) ?

Sabemos que $G(z) = P(z)/\tilde{H}(\lambda, s, z)$, en el segundo paso, es una ción racional cuyas raíces son las mismas que las de P(z) y que tiende al factor li- $z - \rho_j$ cuando $\lambda \to \infty$. Es aquí donde nos percatamos de la segunda razón haber elegido un nuevo origen en el paso anterior, pues una mejor aproximación a ρ_j la obtenemos aplicando a la función G(z) el método de Newton-Raphson lo único que requiere es una primera aproximación s lo suficientemente cercana para garantizarnos convergencia cuadrática hacia dicha raíz. Así pues, tendríamos

$$s_{\lambda+1} = s - \frac{P(s)/\tilde{H}(\lambda, s, s)}{[P(z)/\tilde{H}(\lambda, s, z)]'_{z=s}} = s - \frac{P(s)}{\tilde{H}(\lambda + 1, s, s)} .$$

anterior nos sugiere principiar el tercer paso con el nuevo origen

$$s_L = s - \frac{P(s)}{\tilde{H}(L,\ s,\ s)} \quad , \qquad (3.48)$$

ecir, calculando $H(L + 1, s_L, z)$; pero no sólo esto, sino que antes de calcular guiente H, obtener una mejor aproximación a ρ_j de manera análoga a como se en (3.48), esto es,

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\tilde{H}(\lambda + 1,\ s_\lambda,\ s_\lambda)} \qquad (3.49)$$

ilizarla como nuevo origen en la siguiente iteración, i.e., calcular $H(\lambda + 2,$ $z)$, $\lambda = L, L + 1, \ldots$. De esta manera la convergencia del algoritmo ta superior a la cuadrática pues no sólo se aprovecha la del método de Newton simultáneamente, la geométrica del proceso

$$\frac{P(z)}{\tilde{H}(\lambda,\ s_{\lambda-1},\ z)} \xrightarrow[\lambda \to \infty]{} z - \rho_j \ . \qquad (3.50)$$

Tan sólo nos resta decir la forma en que daremos por terminado último paso. El criterio que se seleccionó nos parece de lo más natural: el eso terminará cuando el límite de aproximación haya sido alcanzado, o dicho en palabras, cuando el valor de $P(z)$ en $z = s_\lambda$ sea de un orden de magnitud me- o igual que el del error de redondeo cometido al calcular $P(s_\lambda)$.

Ya implementado, la manera de aprovechar este algoritmo es la si- ite: los ceros se calculan uno por uno, en orden creciente de magnitud y las s de multiplicidad m_i son determinadas m_i veces; una vez hallada una raíz, el polino- se deflaciona y vuelve a aplicarse el mismo procedimiento al polinomio deflacionado, de anera que P en el algoritmo anterior nos representa al polinomio original o a uno de o menor (ref 10).

3.3 EFICIENCIA DEL ALGORITMO

3.3.1 Ventajas del Algoritmo y Resultados Numéricos. El algoritmo descrito en la segunda sección fue exhaustivamente probado por nosotros mediante una rutina en FORTRAN debida a Jenkins y a Traub (ref 11); tan sólo se le hicieron ligeras modificaciones para adaptarla al sistema BORROUGHS 6700 del CIMASS y se le agregaron un programa principal y dos subrutinas más, una que nos diera el polinomio reconstruido a partir de las raíces encontradas –para tener una prueba de la eficiencia del algoritmo– y una que generara números aleatorios para obtener polinomios arbitrarios y resolverlos. El listado del programa completo, excepto su última subrutina, aparece al final de esta tercera sección. Cabe señalar, además, que aunque nosotros sólo probamos el programa para grados del 3 al 49, el mismo puede modificarse en sus dimensiones para resolver polinomios de grado aun mayor.

La tremenda generalidad del algoritmo queda de manifiesto en las ventajas que posee y que a continuación mencionamos (ref 10).

1. Converge para cualquier distribución de ceros.

2. Las raíces se calculan en orden creciente de magnitud, lo que evita la inestabilidad que tiene lugar cuando el polinomio se deflaciona con un cero grande.

3. El tercer paso es un proceso iterativo, teniendo por consiguiente las propiedades de estabilidad de los métodos iterativos.

4. Pocas decisiones críticas son hechas por el programa que implementa el algoritmo.

5. El algoritmo es rápido para toda distribución de ceros.

6. Las traslaciones se incorporan en el algoritmo mismo de una manera estable y natural, evitándose con ello el problema de los módulos iguales y acelerándose la convergencia.

Por su parte, el programa tiene la gran ventaja de no requerir más datos que los coeficientes del polinomio a resolver, pues una serie de constantes requeridas se proporcionan dentro de una subrutina y la duración de los pasos dos y tres, que varía con cada problema, es elegida por el mecanismo del programa mismo.

La elasticidad del algoritmo nos permitió resolver una gran variedad de polinomios, la gran mayoría de ellos aleatorios con coeficientes complejos. Dentro de los especialmente construidos por nosotros o tomados de otra referencia, se encuentran los siguientes:

1) Polinomio complejo de grado 9 que tiene por raíces a i, i, -1, $1 + i$, $-1 + i$, $-1 + i$, $2i$, $2i$, $2i$; es decir tres de módulo 1, tres de módulo $\sqrt{2}$ y tres de módulo 2.

$$P(z) = z^9 + (2 - 11i)z^8 - (52 + 21i)z^7 - (96 +$$
$$- 137i)z^6 + (215 + 251i)z^5 + (412 +$$
$$- 190i)z^4 - (58 + 434i)z^3 - (284 +$$
$$+ 52i)z^2 - (56 - 104i)z + (16 + 16i).$$

Los resultados obtenidos nos permitieron apreciar una exactitud de por lo menos 11 cifras.

2) Queriendo comprobar las ventajas del algoritmo en lo que se refiere a raíces múltiples y del mismo módulo, metimos, en primer lugar, un polinomio real

de grado 10 cuyas raíces, aunque simples, eran todas de módulo $(0.01)^{1/10}$, es decir,

$$P(z) = z^{10} - 0.01.$$

Enseguida metimos polinomios que elevasen la multiplicidad de las raíces de este polinomio a 2, 3 y 4, respectivamente, i.e., corrimos en sucesión

$$P(z) = (z^{10} - 0.01)^2,$$

$$P(z) = (z^{10} - 0.01)^3,$$

$$P(z) = (z^{10} - 0.01)^4.$$

Los polinomios reconstruidos nos permitieron estimar una aproximación que va desde las 16 cifras –cuando menos– para el primer polinomio hasta una de todavía 5 cifras –por lo menos– para el último, pasando por las de 12 y 9 cifras, respectivamente, para los polinomios intermedios.

3) Para probar, independientemente, altas multiplicidades metimos el polinomio real de grado 10 que tiene por única raíz a 1, i.e.,

$$P(z) = (z - 1)^{10}.$$

Como los resultados fueron óptimos (16 cifras de aproximación, cuando menos), probamos con el polinomio real de grado 20 que duplica la multiplicidad del cero del polinomio anterior, esto es,

$$P(z) = (z - 1)^{20}$$

y los resultados fueron igualmente satisfactorios.

4) Polinomio real de grado 12 cuya dificultad estriba en la alta multiplicidad del módulo 2 en sus raíces.

$$P(z) = (z^6 - 2^6)\,(z + 2)\,(z^2 + 3^2)\,(z - 1)^3.$$

Se obtuvo una aproximación de cuando menos 8 cifras.

5) Polinomio real de cuarto grado cuyos ceros son 9, 10, 1000 y 1001 y de los cuales se desprende que la dificultad para resolverlo radica en que el coeficiente de separación es 0.001 para las dos raíces mayores.

$$P(z) = z^4 - 2020\,z^3 + 1{,}039{,}109\,z^2 - 19{,}199{,}090\,z + 90{,}090{,}000.$$

Los resultados obtenidos fueron los siguientes:

ENTRADA

N = 4

COEFICIENTES

1.0000000000000000D	00000	0.0000000000000000D	00000
−2.0200000000000000D	00003	0.0000000000000000D	00000
1.0391090000000000D	00006	0.0000000000000000D	00000
−1.9199090000000000D	00007	0.0000000000000000D	00000
9.0090000000000000D	00007	0.0000000000000000D	00000

SALIDA

RAICES

9.0000000000000000D	00000	−4.9214859081402417D	−00021
1.0000000000000000D	00001	4.9314283039142428D	−00021
1.0000000000000000D	00003	5.8001939194685233D	−00017
1.0010000000000000D	00003	−5.8001949137081007D	−00017

Es decir, se aprecia una exactitud de cuando menos 16 cifras.

6) Polinomio real de grado 20 con sólo módulos dobles y cuyo coeficiente de separación más pequeño es 0.0246.

$$P(z) = z^{20} - 20\,z^{18} + 170\,z^{16} - 800\,z^{14} + 2275\,z^{12} - 4004\,z^{10} +$$
$$+ 4290\,z^{8} - 2640\,z^{6} + 825\,z^{4} - 100\,z^{2} + 2.$$

Se obtuvo una aproximación de por lo menos 15 cifras.

7) Polinomio ciclotómico real de grado 9 cuyas raíces son $(1 \pm i)\sqrt{1/2}$, $\pm i$, $(-1 \pm i)\sqrt{1/2}$, -1, 0, 0, es decir, una doble y siete de módulo 1.

$$P(z) = z^9 + z^8 + z^7 + z^6 + z^5 + z^4 + z^3 + z^2 = z^2 \frac{z^8 - 1}{z - 1}\,.$$

El polinomio reconstruido nos permitió estimar una aproximación mínima de 16 cifras.

NOTA. Los siguientes dos polinomios son típicos por la extrema dificultad que encierra su resolución (ref 16).

8) Polinomio real de grado 16 que posee únicamente pares de raíces complejas conjugadas en el semi-plano izquierdo.

$$P(z) = 2.03253121\,z^{16} + 3.4356048\,z^{15} +$$
$$+ 25.1783048\,z^{14} + 37.651096\,z^{13} +$$
$$+ 128.218748\,z^{12} + 166.44768\,z^{11} +$$
$$+ 345.07256\,z^{10} + 378.908\,z^{9} +$$
$$+ 524.327\,z^{8} + 468.88\,z^{7} + 443.576\,z^{6} +$$
$$+ 304.08\,z^{5} + 190.68\,z^{4} + 89.6\,z^{3} +$$
$$+ 32.8\,z^{2} + 8\,z + 1.$$

Este polinomio no representó problema alguno para el algoritmo descrito en este trabajo, pues lo resolvió con una aproximación mínima de 13 cifras.

9) Polinomio real de grado 16 cuya dificultad radica en el hecho de tener únicamente raíces complejas conjugadas y de módulos cercanos: cinco pares de raíces se juntan alrededor del eje imaginario.

$$\begin{aligned} P(z) = {} & 1250162561\, z^{16} + 385455882\, z^{15} + \\ & + 845947696\, z^{14} + 240775148\, z^{13} + \\ & + 247926664\, z^{12} + 64249356\, z^{11} + \\ & + 41018752\, z^{10} + 9490840\, z^{9} + 4178260\, z^{8} + \\ & + 837860\, z^{7} + 267232\, z^{6} + 44184\, z^{5} + \\ & + 10416\, z^{4} + 1288\, z^{3} + 224\, z^{2} + 16\, z + 2. \end{aligned}$$

Tampoco la dificultad intrínseca de este polinomio representó mucho problema para el algoritmo ya que lo resolvió con un mínimo de 8 cifras de aproximación.

Sin embargo, como dijimos anteriormente, la gran mayoría de los polinomios resueltos por el algoritmo discutido en este trabajo fueron aleatorios y con coeficientes complejos. En base a ello, elaboramos la siguiente tabla en la cual incluimos el tiempo de ejecución promedio para resolver un polinomio de grado n ($3 \leqslant n \leqslant 49$) y la aproximación estimada a partir de los polinomios reconstruidos.

Con los tiempos dados en la cuarta columna de la tabla siguiente elaboramos la gráfica que aparece posteriormente y que nos permite apreciar mejor el tiempo promedio que tarda el programa en resolver un polinomio de grado n ($3 \leqslant n \leqslant 49$) y compararlo con el que tarda para resolver polinomios de otros grados.

TABLA 3.1. TABLA DE POLINOMIOS ALEATORIOS RESUELTOS POR EL ALGORITMO ITERACION DE TRES PASOS DE DESPLAZAMIENTO VARIABLE

Grado n	Número de polinomios resueltos	Tiempo de ejecución total (segs.)	Tiempo de ejecución promedio (segs.)	Mínima aproximación estimada
3	9	3.32	0.37	11 cifras
4	10	5.38	0.538	11 cifras
5	10	6.74	0.674	16 cifras
6	10	8.63	0.863	9 cifras
7	10	11.29	1.129	9 cifras
8	10	13.42	1.342	7 cifras
9	10	15.52	1.552	9 cifras
10	10	18.95	1.895	15 cifras
11	9	20.808	2.312	9 cifras
12	10	25.537	2.5537	8 cifras
13	10	29.072	2.9072	9 cifras
14	10	30.947	3.0947	8 cifras
15	10	35.478	3.5478	15 cifras
16	10	38.908	3.8908	8 cifras
17	10	43.700	4.3700	7 cifras
18	10	49.919	4.9919	7 cifras
19	10	51.321	5.1321	10 cifras
20	10	57.121	5.7121	15 cifras
21	10	63.651	6.3651	10 cifras
22	9	59.513	6.6126	9 cifras
23	10	71.905	7.1905	9 cifras
24	10	76.590	7.6590	9 cifras
25	10	80.653	8.0653	7 cifras
26	10	89.388	8.9388	7 cifras
27	9	84.411	9.3790	10 cifras
28	10	100.030	10.0030	9 cifras
29	10	106.469	10.6469	7 cifras
30	10	112.794	11.2794	11 cifras
31	10	117.992	11.7992	7 cifras
32	9	116.332	12.9258	9 cifras
33	8	109.407	13.6759	9 cifras
34	9	124.184	13.7982	9 cifras
35	10	155.989	15.5989	8 cifras
36	10	155.595	15.5595	7 cifras
37	9	154.592	17.1769	10 cifras
38	10	184.586	18.4586	10 cifras
39	10	193.975	19.3975	9 cifras
40	10	198.467	19.8467	10 cifras
41	10	218.711	21.8711	9 cifras
42	10	226.344	22.6344	9 cifras
43	10	227.469	22.7469	8 cifras
44	10	249.235	24.9235	6 cifras
45	9	215.825	23.9805	7 cifras
46	10	250.702	25.0702	10 cifras
47	10	263.388	26.3388	7 cifras
48	10	275.618	27.5618	10 cifras
49	10	280.666	28.0666	8 cifras

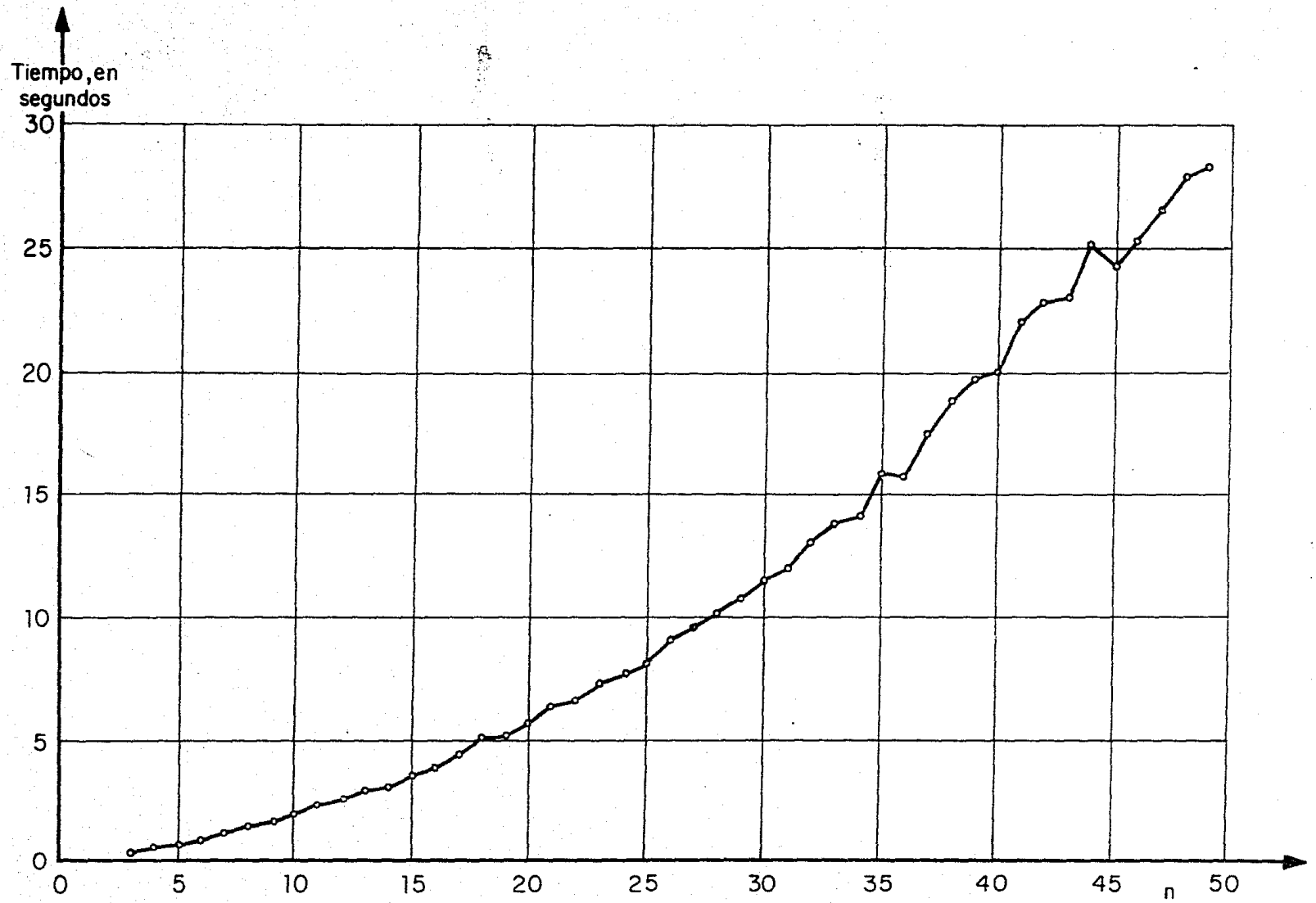


Fig 3.1

Este programa fue probado por un total de 629 polinomios de grados 3 al 49, de los cuales 609 fueron aleatorios con coeficientes complejos y 20 especialmente construidos por nosotros o tomados de (ref 16), habiéndose puesto especial empeño en que estos últimos resultaran realmente difíciles de resolver. Creemos que los resultados no pudieron haber sido mejores: los 629 polinomios fueron resueltos y con las magníficas aproximaciones antes señaladas.

Para concluir la tercera sección de este capítulo presentamos el listado del programa en FORTRAN que se utilizó.

3.3.2. Programa en FORTRAN IV para Polinomios Complejos

```
                                                                  COMPILACION EN FORTRAN B6700
                                                                                NIVEL II,02,04
                                                          NOVIEMBRE      MARTES/07/72  12:43 PM
                                                                        FIB IS 0007 LONG
                                                                  START OF SEGMENT 002
FILE  5= LECTOR, UNIT = READER                                                          C  002:0000:0
C     ESTE PROGRAMA NOS CALCULA TODOS LOS CEROS DE UN POLINOMIO COMPLEJO                 C  002:0000:0
C     MEDIANTE EL ALGORITMO COMPLEJO DE TRES-ESTADOS ESTUDIADO POR JEN =               C  002:0000:0
C     KINS Y TRAUB, EL PROGRAMA CALCULA LOS CEROS UNO A LA VEZ Y EN OR =               C  002:0000:0
C     DEN CRECIENTE DE MODULO, DEFLACIONANDO EL POLINOMIO A UNO DE GRADO               C  002:0000:0
C     MENOR, EL PROGRAMA ES EXTREMADAMENTE RAPIDO SIENDO ESTA RAPIDEZ =                C  002:0000:0
C     INDEPENDIENTE DE LA DISTRIBUCION DE LOS CEROS,                                   C  002:0000:0
C                                                                                      C  002:0000:0
      DOUBLE PRECISION OPR(50),OPI(50),ZEROR(49),ZEROI(49),XR(100),XI(10               C  002:0000:0
     *0),YR(100)                                                                       C  002:0000:0
      LOGICAL FAIL                                                                     C  002:0000:0
      INTEGER DEGREE                                                                   C  002:0000:0
10000 READ(5,1000,END=62) DEGREE                                                       C  002:0000:0
      NI = DEGREE + 1                                                                  C  002:0000:0
      READ(5,1001) (OPR(I), I= 1,NI), (OPI(I), I = 1,NI)                               C  002:0000:2
      PRINT 2000, DEGREE, (OPR(I), OPI(I), I = 1,NI)                                   C  002:0023:2
                                                                        FIB IS 0005 LONG
C     INICIALIZACION                                                                   C  002:0038:2
C                                                                                      C  002:0038:2
      CALL CPOLY(OPR,OPI,DEGREE,ZEROR,ZEROI,FAIL)                                      C  002:0038:2
      PRINT 2001, (ZEROR(I), ZEROI(I), I = 1,DEGREE)                                   C  002:003C:4
      CALL POLORI(ZEROR,ZEROI,OPR(1),OPI(1),XR,XI,YR,DEGREE)                           C  002:004F:5
      PRINT 2002, (XR(I),XI(I), I=1,NI)                                                C  002:0056:0
      TEJ = TIME(2)/60.0                                                               C  002:0069:2
      TES = TIME(3)/60.0                                                               C  002:006B:2
      PRINT 222, TEJ,TES                                                               C  002:006D:2
      GO TO 10000                                                                      C  002:007B:0
   62 CALL EXIT                                                                        C  002:007B:3
C     FORMATOS                                                                         C  002:007C:2
C                                                                                      C  002:007C:2
 1000 FORMAT(I6)                                                                       C  002:007C:2
 1001 FORMAT(8D10.0)                                                                   C  002:007C:2
 2000 FORMAT(1H1,8H ENTRADA//1X,4HN = ,I3//1X,12HCOEFICIENTES///                       C  002:007C:2
     *(1X,2(1PD26.16,4X))///)                                                          C  002:007C:2
 2001 FORMAT(1HO,7H SALIDA//1X,6HRAICES///(1X,2(1PD26.16,4X))///)                      C  002:007C:2
 2002 FORMAT(1HO,22HPOLINOMIO-RECONSTRUIDO///(1X,2(1PD26.16,4X))////)                  C  002:007C:2
  222 FORMAT(1HO,21HTIEMPO DE EJECUCION =,F5.2,5HSEGS.,3X,                             C  002:007C:2
     *29HTIEMPO DE ENTRADA Y SALIDA =,F5.2,5HSEGS.)                                    C  002:007C:2
      END                                                                              C  002:007C:2
                                                        REENTRANT FORMAT SEGMENT 003 IS 0003 LONG
                                                          NONREENTRANT FORMAT ARRAY IS 0030 LONG
      002:0098:4  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
      002:009D:0  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
      002:009E:2  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
      002:009F:4  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
      002:00A1:0  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
      002:00A4:1  IS THE LOCATION FOR EXCEPTIONAL ACTION ON THE I/O STATEMENT AT
                                                                  SEGMENT 002 IS 00A5 LONG
```

```
      SUBROUTINE CPOLY(OPR,OPI,DEGREE,ZEROR,ZEROI,FAIL)
C     ENCUENTRA LOS CEROS DE UN POLINOMIO COMPLEJO.
C     OPR,OPI - VECTORES EN DOBLE-PRECISION DE LAS PARTES
C     REAL E IMAGINARIA DE LOS COEFICIENTES EN
C     ORDEN DECRECIENTE DE POTENCIAS.
C     DEGREE - GRADO ENTERO DEL POLINOMIO.
C     ZEROR,ZEROI - VECTORES DE SALIDA EN DOBLE PRECISION DE
C     LAS PARTES REAL E IMAGINARIA DE LOS CEROS.
C     FAIL - PARAMETRO LOGICO DE SALIDA, VERDADERO SOLO SI
C     EL COEFICIENTE INICIAL ES CERO O SI CPOLY
C     HA CALCULADO MENOS DE DEGREE CEROS.
C     EL PROGRAMA HA SIDO ESCRITO PARA REDUCIR LA POSIBILIDAD DE OVERFLO W.
C     SI ESTO OCURRE, EXISTE TODAVIA LA POSIBILIDAD DE QUE
C     EL METODO TRABAJE SI LA CANTIDAD CAUSANTE DEL OVERFLOW SE
C     REEMPLAZA POR UN NUMERO GRANDE.
C     COMMON AREA.
      COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,
     *SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,IFIN,NN
      DOUBLE PRECISION SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,IFIN,
     *PR(50),PI(50),HR(50),HI(50),QPR(50),QPI(50),QHR(50),
     *QHI(50),SHR(50),SHI(50),OPR(50),OPI(50),ZEROR(49),ZEROI(49)
C     PARA CAMBIAR EL TAMANO DE LOS POLINOMIOS QUE PUEDEN SER RESUELTOS,
C     REEMPLAZAR LA DIMENSION DE LOS ARREGLOS EN EL AREA COMUN.
      DOUBLE PRECISION XX,YY,COSR,SINR,SMALNO,BASE,XXX,ZR,ZI,BND,
     *CMOD,SCALE,CAUCHY,DSQRT
      LOGICAL FAIL, CONV
      INTEGER DEGREE,CNT1,CNT2
C     INICIALIZACION DE CONSTANTES.
      CALL MCON(ETA,IFIN,SMALNO,BASE)
      ARE = ETA
      MRE = 2.0D0*DSQRT(2.0D0)*ETA
      XX = .70710678
      YY = - XX
      COSR = - .060756474
      SINR = .99756405
      FAIL = .FALSE.
      NN = DEGREE + 1
C     EL ALGORITMO FALLA SI EL COEFICIENTE INICIAL ES CERO.
      IF(OPR(1).NE.0.0D0.OR.OPI(1).NE.0.0D0) GO TO 10
      FAIL = .TRUE.
      RETURN
C     REMOVER  LOS CEROS DEL ORIGEN SI ES QUE EXISTE ALGUNO.
   10 IF(OPR(NN).NE.0.0D0.OR.OPI(NN).NE.0.0D0) GO TO 20
      IDNN2 = DEGREE - NN + 2
      ZEROR(IDNN2)=0.0D0
      ZEROI(IDNN2) = 0.0D0
      NN = NN - 1
      GO TO 10
C     HACER UNA COPIA DE LOS COEFICIENTES.
   20 DO 30 I = 1,NN
      PR(I) = OPR(I)
      PI(I) = OPI(I)
      SHR(I) = CMOD(PR(I),PI(I))
   30 CONTINUE
C     GRADUAR EL POLINOMIO.
      BND = SCALE(NN,SHR,ETA,IFIN,SMALNO,BASE)
      IF(BND.EQ.1.0D0) GO TO 40
      DO 35 I = 1,NN
      PR(I) = BND*PR(I)
```

```
START OF SEGMENT 005
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
   C  005:0000:0
31 C  005:0003:4
   C  005:0005:5
   C  005:000A:2
   C  005:000C:4
   C  005:000D:2
   C  005:000F:5
   C  005:0011:4
   C  005:0012:2
   C  005:0014:2
   C  005:0014:2
   C  005:0017:4
   C  005:0018:2
   C  005:0018:5
   C  005:001B:5
   C  005:001F:1
   C  005:0021:5
   C  005:0024:0
   C  005:0026:1
   C  005:0028:1
   C  005:0028:4
   C  005:002B:4
   C  005:002A:0
   C  005:002D:0
   C  005:0030:1
   C  005:0035:2
   C  005:0038:1
   C  005:003B:1
   C  005:003E:2
   C  005:003F:3
   C  005:0041:0
```

```
      PI(I) = BND*PI(I)                                                 C  005:0044:0
   35 CONTINUE                                                          C  005:0047:2
C     PRINCIPIAR EL ALGORITMO PARA ALGUN CERO.                          C  005:004A:1
   40 IF(NN.GT.2) GO TO 50                                              C  005:004A:1
C     CALCULAR EL CERO FINAL Y REGRESAR.                                C  005:004C:1
      CALL CDIVID(-PR(2),-PI(2),PR(1),PI(1),ZEROR(DEGREE),              C  005:004C:1
     *ZEROI(DEGREE))                                                    C  005:0054:3
      RETURN                                                            C  005:0056:3
C     CALCULAR BND, UNA COTA INFERIOR PARA LOS MODULOS DE LOS CEROS.    C  005:0057:0
   50 DO 60 I = 1,NN                                                    C  005:0057:0
      SHR(I) = CMOD(PR(I),PI(I))                                        C  005:0058:0
   60 CONTINUE                                                          C  005:005D:1
      BND = CAUCHY(NN,SHR,SHI)                                          C  005:0060:0
C     LOOP EXTERIOR PARA CONTROLAR 2 PASES MAYORES CON DISTINTAS SUCESIONES  005:0064:1
C     DE CAMBIOS.                                                       C  005:0064:1
      DO 100 CNT1 = 1,2                                                 C  005:0064:1
C     PRIMER ESTADO DE CALCULO, NINGUN CAMBIO.                          C  005:0065:0
      CALL NOSHFT(5)                                                    C  005:0065:0
C     LOOP INTERNO PARA SELECCIONAR UN CAMBIO.                          C  005:0066:0
      DO 90 CNT2 = 1,9                                                  C  005:0066:0
C     SE ESCOGE UN CAMBIO DE MODULO BND Y AMPLITUD ROTADA               C  005:0067:0
C     94 GRADOS DEL CAMBIO ANTERIOR.                                    C  005:0067:0
      XXX = COSR*XX -SINR*YY                                            C  005:0067:0
      YY = SINR*XX + COSR*YY                                            C  005:0069:2
      XX = XXX                                                          C  005:006B:4
      SR = BND*XX                                                       C  005:006C:3
      SI = BND*YY                                                       C  005:006E:3
C     SEGUNDO ESTADO DE CALCULO, CAMBIO FIJO.                           C  005:0070:3
      CALL FXSHFT(10*CNT2,ZR,ZI,CONV)                                   C  005:0070:3
      IF(.NOT.CONV) GO TO 80                                            C  005:0073:3
C     EL SEGUNDO ESTADO SALTA DIRECTAMENTE AL TERCER ESTADO DE LA ITERACION.  005:0074:3
C    SI SE TIENE EXITO EL CERO SE GUARDA Y EL POLINOMIO SE DEFLACIONA.   C  005:0074:3
      IDNN2 = DEGREE - NN + 2                                           C  005:0074:3
      ZEROR(IDNN2) = ZR                                                 C  005:0077:1
      ZEROI(IDNN2) = ZI                                                 C  005:0079:2
      NN = NN - 1                                                       C  005:007B:3
      DO 70 I = 1,NN                                                    C  005:007D:3
      PR(I) = QPR(I)                                                    C  005:007F:0
      PI(I) = QPI(I)                                                    C  005:0081:4
   70 CONTINUE                                                          C  005:0084:3
      GO TO 40                                                          C  005:0087:2
   80 CONTINUE                                                          C  005:0087:5
C     SI SE FRACASA EN LA ITERACION SE ESCOGE OTRO CAMBIO.              C  005:0087:5
   90 CONTINUE                                                          C  005:0087:5
C     SI FALLAN 9 CAMBIOS, EL LOOP EXTERIOR SE REPITE CON OTRA          C  005:008A:0
C     SUCESION DE CAMBIOS.                                              C  005:008A:0
  100 CONTINUE                                                          C  005:008A:0
C     EL METODO HA FALLADO EN LOS PASES MAYORES.                        C  005:008C:1
C     RETURN EMPTY HANDED.                                              C  005:008C:1
      FAIL = .TRUE.                                                     C  005:008C:1
      RETURN                                                            C  005:008C:5
      END                                                               C  005:008D:2
SEGMENT 005 IS 009D LONG
```

```
      SUBROUTINE NOSHFT(L1)
C     CALCULA LA DERIVADA DEL POLINOMIO COMO EL POLINOMIO
C     INICIAL H Y CALCULA LOS POLINOMIOS H SIN CAMBIOS.
C     COMMON AREA.
      COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,
     *SP,SI,TR,TI,PVR,PVI,ARE,HRE,ETA,IFIN,NN
      DOUBLE PRECISION SR,SI,TR,TI,PVR,PVI,ARE,HRE,ETA,IFIN,
     *PR(50),PI(50),HR(50),HI(50),QPR(50),QPI(50),QHR(50),
     *QHI(50),SHR(50),SHI(50)
      DOUBLE PRECISION XNI,T1,T2,CMOD
      N = NN - 1
      NM1 = N - 1
      DO 10 I = 1,N
      XNI = NN - I
      HR(I) = XNI*PR(I)/FLOAT(N)
      HI(I) = XNI*PI(I)/FLOAT(N)
   10 CONTINUE
      DO 50 JJ = 1,L1
      IF(CMOD(HR(N),HI(N)).LE.ETA*10.0D0*CMOD(PR(N),PI(N)))
     *GO TO 30
      CALL CDIVID(-PR(NN),-PI(NN),HR(N),HI(N),TR,TI)
      DO 20 I = 1,NM1
      J = NN - I
      T1 = HR(J-1)
      T2 = HI(J-1)
      HR(J) = TR*T1 - TI*T2 + PR(J)
      HI(J) = TR*T2 + TI*T1 + PI(J)
   20 CONTINUE
      HR(1) = PR(1)
      HI(1) = PI(1)
      GO TO 50
C     SI EL TERMINO CONSTANTE ES ESENCIALMENTE CERO, CAMBIAR H COEFICIENTES.
   30 DO 40 I = 1,NM1
      J = NN - I
      HR(J) = HR(J-1)
      HI(J) = HI(J-1)
   40 CONTINUE
      HR(1) = 0.0D0
      HI(1) = 0.0D0
   50 CONTINUE
      RETURN
      END
```

```
START OF SEGMENT 007
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0000:0
C  007:0002:0
C  007:0003:0
C  007:0004:0
C  007:0006:1
C  007:0009:5
C  007:000D:4
C  007:000F:5
C  007:0011:0
C  007:001A:5
C  007:001A:5
C  007:0027:0
C  007:0028:0
C  007:002A:1
C  007:002C:0
C  007:002D:5
C  007:0033:5
C  007:003A:0
C  007:003C:1
C  007:003D:5
C  007:003F:4
C  007:0040:1
C  007:0040:1
C  007:0041:0
C  007:0043:1
C  007:0046:0
C  007:0048:5
C  007:004B:0
C  007:004C:2
C  007:004D:4
C  007:004F:5
C  007:0050:2
SEGMENT 007 IS 0058 LONG
```

```
                                                                        START OF SEGMENT 008
      SUBROUTINE FXSHFT(L2,ZR,ZI,CONV)                                  C  008:0000:0
C     CALCULA L2 POLINOMIOS H DE CAMBIO FIJO Y PRUEBA PARA              C  008:0000:0
C     CONVERGENCIA.                                                     C  008:0000:0
C     INICIA UNA ITERACION DE CAMBIO VARIABLE Y REGRESA CON EL          C  008:0000:0
C     CERO APROXIMADO SI SE TIENE EXITO.                                C  008:0000:0
C     L2 = LIMITE DE PASOS DE CAMBIO FIJO.                              C  008:0000:0
C     ZR,ZI = CERO APROXIMADO SI CONV ES .TRUE.                         C  008:0000:0
C     CONV = VARIABLE LOGICA QUE INDICA CONVERGENCIA EN EL ESTADO 3 DE L A  C  008:0000:0
C     ITERACION.                                                        C  008:0000:0
C     COMMON AREA.                                                      C  008:0000:0
      COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,                C  008:0000:0
     *SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,IFIN,NN                           C  008:0000:0
      DOUBLE PRECISION SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,IFIN,            C  008:0000:0
     *PR(50),PI(50),HR(50),HI(50),QPR(50),QPI(50),QHR(50),              C  008:0000:0
     *QHI(50),SHR(50),SHI(50)                                           C  008:0000:0
      DOUBLE PRECISION ZR,ZI,OTR,OTI,SVSR,SVSI,CMOD                     C  008:0000:0
      LOGICAL CONV,TEST,PASD,BOOL                                       C  008:0000:0
      N = NN - 1                                                        C  008:0000:0
C     CALCULAR P EN S.                                                  C  008:0002:0
      CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)                       C  008:0002:0
      TEST = .TRUE.                                                     C  008:000A:5
      PASD = .FALSE.                                                    C  008:000B:3
C     CALCULAR PRIMERO T = -P(S)/H(S).                                  C  008:000C:1
      CALL CALCT(BOOL)                                                  C  008:000C:1
C     LOOP PRINCIPAL PARA UN PASO1 SEGUNDO ESTADO.                      C  008:000D:2
      DO 50 J = 1,L2                                                    C  008:000D:2
      OTR = TR                                                          C  008:000E:0
      OTI = TI                                                          C  008:000F:3
C     CALCULAR EL SIGUIENTE POLINOMIO H Y EL NUEVO T.                   C  008:0011:0
      CALL NEXTH(BOOL)                                                  C  008:0011:0
      CALL CALCT(BOOL)                                                  C  008:0012:1
      ZR = SR + TR                                                      C  008:0013:2
      ZI = SI + TI                                                      C  008:0016:0
C     PROBAR PARA CONVERGENCIA AL MENOS QUE EL ESTADO 3 HAYA FALLADO EST A VEZ  O  C  008:0018:4
C     ESTE ES EL ULTIMO POLINOMIO H.                                    C  008:0018:4
      IF(BOOL.OR..NOT.TEST.OR.J.EQ.L2) GO TO 50                         C  008:001B:4
      IF(CMOD(TR-OTR,TI-OTI).GE..5D0*CMOD(ZR,ZI)) GO TO 40              C  008:001B:1
      IF(.NOT.PASD) GO TO 30                                            C  008:0024:4
C     LA PRUEBA DEBIL DE CONVERGENCIA HA SIDO PASADA DOS VECES, COMENZAR  EL  C  008:0025:4
C     TERCER ESTADO DE LA ITERACION, DESPUES DE SALVAR AL POLINOMIO H AC TUAL  C  008:0025:4
C     Y CAMBIAR.                                                        C  008:0025:4
      DO 10 I = 1,N                                                     C  008:0025:4
      SHR(I) = HR(I)                                                    C  008:0027:0
      SHI(I) = HI(I)                                                    C  008:002A:0
   10 CONTINUE                                                          C  008:002D:0
      SVSR = SR                                                         C  008:002F:1
      SVSI = SI                                                         C  008:0030:4
      CALL VRSHFT(10,ZR,ZI,CONV)                                        C  008:0032:1
      IF(CONV) RETURN                                                   C  008:0034:4
C     LA ITERACION FALLO LA CONVERGENCIA. DEJAR LA PRUEBA Y RESTAURAR H, S,PV,Y T  C  008:0036:0
      TEST = .FALSE.                                                    C  008:0036:0
      DO 20 I = 1,N                                                     C  008:0036:4
      HR(I) = SHR(I)                                                    C  008:0038:0
      HI(I) = SHI(I)                                                    C  008:003B:0
   20 CONTINUE                                                          C  008:003E:0
      SR = SVSR                                                         C  008:0040:1
      SI = SVSI                                                         C  008:0041:4
      CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)                       C  008:0043:1
      CALL CALCT(BOOL)                                                  C  008:004C:0
```

```
      GO TO 50
   30 PASO = .TRUE.
      GO TO 50
   40 PASO = .FALSE.
   50 CONTINUE
C     REALICEMOS UNA ITERACION CON EL POLINOMIO H FINAL DEL SEGUNDO ESTA DO,
      CALL VRSHFT(10,ZR,ZI,CONV)
      RETURN
      END
```

```
C  008:004D:1
C  008:004D:4
C  008:004E:2
C  008:004E:5
C  008:004F:3
C  006:0051:4
C  008:0051:4
C  008:0054:1
C  008:0054:4
SEGMENT 008 IS 0067 LONG
```

```
      SUBROUTINE VRSHFT(L3,ZR,ZI,CONV)
C     LLEVA A CABO EL TERCER ESTADO DE LA ITERACION.
C     L3 = LIMITE DE PASOS EN EL ESTADO 3.
C     ZR,ZI = AL PRINCIPIO CONTIENE A LA ITERACION INICIAL, SI LA
C     ITERACION CONVERGE CONTIENE A LA ITERACION FINAL
C     A LA SALIDA.
C     CONV = .TRUE. SI LA ITERACION CONVERGE.
C     COMMON AREA.
      COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,
     *SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,IFIN,NN
      DOUBLE PRECISION SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,IFIN,
     *PR(50),PI(50),HR(50),HI(50),QPR(50),QPI(50),QHR(50),
     *QHI(50),SHR(50),SHI(50)
      DOUBLE PRECISION ZR,ZI,HP,MS,OHP,RELSTP,R1,R2,CMOD,DSQRT,ERREV,TP
      LOGICAL CONV,B,BOOL
      CONV = .FALSE.
      B = .FALSE.
      SR = ZR
      SI = ZI
C     LOOP PRINCIPAL PARA EL ESTADO TRES.
      DO 60 I = 1,L3
C     CALCULAR P EN S Y PROBAR CONVERGENCIA.
      CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)
      HP = CMOD(PVR,PVI)
      MS = CMOD(SR,SI)
      IF(HP.GT.20.0D0*ERREV*MS-QPR,QPI,MS,HP,ARE,MRE)) GO TO 10
C     EL VALOR DEL POLINOMIO ES MENOR EN VALOR QUE UNA COTA EN EL ERROR
C     AL CALCULAR P, TERMINA LA ITERACION.
      CONV = .TRUE.
      ZR = SR
      ZI = SI
      RETURN
   10 IF(I.EQ.1) GO TO 40
      IF(B.OR.HP.LT.OHP.OR.RELSTP.GE..05D0) GO TO 30
C     LA ITERACION SE HA DETENIDO, PROBABLE ACUMULACION DE CEROS, HACER 5  PASOS
C     DE CAMBIO FIJO DENTRO DEL PUNTO DE ACUMULACION PARA FORZAR A UN CERO A
C     DOMINAR.
      TP = RELSTP
      B = .TRUE.
      IF(RELSTP.LT.ETA) TP = ETA
      R1 = DSQRT(TP)
      R2 = SR*(1.0D0 + R1) - SI*R1
      SI = SR*R1 + SI*(1.0D0+R1)
      SR = R2
      CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)
      DO 20 J = 1,5
      CALL CALCT(BOOL)
      CALL NEXTH(BOOL)
   20 CONTINUE
      OHP = IFIN
      GO TO 50
C     SALIDA SI EL VALOR DEL POLINOMIO CRECE SIGNIFICATIVAMENTE.
   30 IF(HP*.100.GT.OHP) RETURN
   40 OHP = HP
C     CALCULAR LA SIGUIENTE ITERACION.
   50 CALL CALCT(BOOL)
      CALL NEXTH(BOOL)
      CALL CALCT(BOOL)
      IF(BOOL) GO TO 60
```

```
START OF SEGMENT 009
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:0
C  009:0000:4
C  009:0001:2
C  009:0002:5
C  009:0004:2
C  009:0004:2
C  009:0005:0
C  009:0005:5
C  009:0011:0
C  009:0014:1
C  009:001C:2
C  009:001C:2
C  009:001C:2
C  009:001D:0
C  009:001E:3
C  009:0020:0
C  009:0020:3
C  009:0021:4
C  009:0028:1
C  009:0028:1
C  009:0028:1
C  009:0028:1
C  009:0029:0
C  009:0029:4
C  009:002D:1
C  009:002E:4
C  009:0033:0
C  009:0037:2
C  009:003B:5
C  009:0041:4
C  009:0043:0
C  009:0044:1
C  009:0045:2
C  009:0047:3
C  009:0049:0
C  009:0049:3
C  009:0049:3
C  009:004F:0
C  009:004F:5
C  009:004F:5
C  009:0051:0
C  009:0052:1
C  009:0053:2
```

```
      RELSTP = CHOD(TR,TI)/CHOD(SR,SI)
      SR = SR + TR
      SI = SI + TI
   60 CONTINUE
      RETURN
      END
```

```
C  009:0054:1
C  009:005A:1
C  009:005C:5
C  009:005F:3
C  009:0061:4
C  009:0062:1
SEGMENT 009 IS 0074 LONG
```

```
      SUBROUTINE CALCT(BOOL)
C     CALCULAR T = -P(S)/H(S).
C     BOOL = LOGICAL, .TRUE. SI H(S) ES ESENCIALMENTE CERO.
C     COMMON AREA.
      COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,
     *SR,SI,TR,TI,PVR,PVI,ARE,HRE,ETA,IFIN,NN
      DOUBLE PRECISION SR,SI,TR,TI,PVR,PVI,ARE,HRE,ETA,IFIN,
     *PR(50),PI(50),HR(50),HI(50),QPR(50),QPI(50),QHR(50),
     *QHI(50),SHR(50),SHI(50)
      DOUBLE PRECISION HVR,HVI,CMOD
      LOGICAL BOOL
      N = NN - 1
C     CALCULAR H(S).
      CALL POLYEV(N,SR,SI,HR,HI,QHR,QHI,HVR,HVI)
      BOOL = CMOD(HVR,HVI).LE.ARE*10.0D0*CMOD(HR(N),HI(N))
      IF(BOOL) GO TO 10
      CALL CDIVID(-PVR,-PVI,HVR,HVI,TR,TI)
      RETURN
   10 TR = 0.0D0
      TI = 0.0D0
      RETURN
      END
```

```
 START OF SEGMENT 00A
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0000:0
C  00A:0002:0
C  00A:0002:0
C  00A:0009:5
C  00A:0011:5
C  00A:0012:2
C  00A:001A:2
C  00A:001A:5
C  00A:001C:2
C  00A:001D:5
C  00A:001E:2
SEGMENT 00A IS 0027 LONG
```

```
      SUBROUTINE NEXTH(BOOL)
C     CALCULA EL SIGUIENTE POLINOMIO CAMBIADO H.
C     BOOL = LOGICA, SI =.TRUE., H(S)=ES=ESENCIALMENTE CERO,
C     COMMON AREA.
      COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,
     *SR,SI,TR,TI,PVR,PVI,ARE,NRE,ETA,IFIN,NN
      DOUBLE PRECISION SR,SI,TR,TI,PVR,PVI,ARE,NRE,ETA,IFIN,
     *PR(50),PI(50),HR(50),HI(50),QPR(50),QPI(50),QHR(50),
     *QHI(50),SHR(50),SHI(50)
      DOUBLE PRECISION T1,T2
      LOGICAL BOOL
      N = NN - 1
      NH1 = N - 1
      IF(BOOL) GO TO 20
      DO 10 J = 2,N
      T1 = QHR(J-1)
      T2 = QHI(J-1)
      HR(J) = TR*T1 - TI*T2 + QPR(J)
      HI(J) = TR*T2 + TI*T1 + QPI(J)
   10 CONTINUE
      HR(1) = QPR(1)
      HI(1) = QPI(1)
      RETURN
C     SI H(S) ES CERO REEMPLAZAR H CON QH.
   20 DO 30 J = 2,N
      HR(J) = QHR(J-1)
      HI(J) = QHI(J-1)
   30 CONTINUE
      HR(1) = 0.0D0
      HI(1) = 0.0D0
      RETURN
      END
```

```
 START OF SEGMENT 00B
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0000:0
C  00B:0002:0
C  00B:0003:0
C  00B:0003:5
C  00B:0005:0
C  00B:0007:0
C  00B:0009:0
C  00B:000F:1
C  00B:0015:2
C  00B:0017:3
C  00B:0019:2
C  00B:001B:1
C  00B:001B:4
C  00B:001B:4
C  00B:001D:0
C  00B:0020:0
C  00B:0023:0
C  00B:0025:1
C  00B:0026:3
C  00B:0027:5
C  00B:0028:2
SEGMENT 00B IS 002F LONG
```

```
      SUBROUTINE POLYEV(NN,SR,SI,PR,PI,QR,QI,PVR,PVI)
C     CALCULA UN POLINOMIO P EN S MEDIANTE LA RECURRENCIA DE HORNER
C     COLOCANDO LAS SUMAS PARCIALES EN Q Y EL VALOR CALCULADO EN PV.
      DOUBLE PRECISION PR(NN),PI(NN),QR(NN),QI(NN),
     *SR,SI,PVR,PVI,T
      QR(1) = PR(1)
      QI(1) = PI(1)
      PVR = QR(1)
      PVI = QI(1)
      DO 10 I = 2,NN
      T = PVR*SR - PVI*SI + PR(I)
      PVI = PVR*SI + PVI*SR + PI(I)
      PVR = T
      QR(I) = PVR
      QI(I) = PVI
   10 CONTINUE
      RETURN
      END
```

```
 START OF SEGMENT 00C
C  00C:0000:0
C  00C:0000:0
C  00C:0000:0
C  00C:0000:0
C  00C:0000:0
C  00C:0000:0
C  00C:0001:5
C  00C:0003:4
C  00C:0005:0
C  00C:0006:2
C  00C:0008:0
C  00C:000C:1
C  00C:0010:2
C  00C:0011:1
C  00C:0013:2
C  00C:0015:3
C  00C:0017:4
C  00C:0018:1
SEGMENT 00C IS 0029 LONG
```

```
      DOUBLE PRECISION FUNCTION ERREV(NN,QR,QI,MS,MP,ARE,MRE)
C     ACOTA EL ERROR EN EL CALCULO DEL POLINOMIO MEDIANTE LA RECURRENCIA
C     DE HORNER.
C     QR,QI = LAS SUMAS PARCIALES.
C     MS = MODULO DEL PUNTO.
C     MP = MODULO DEL VALOR DEL POLINOMIO.
C     ARE,MRE = COTAS DEL ERROR EN LA SUMA Y LA MULTIPLICACION COMPLEJAS .
      DOUBLE PRECISION QR(NN),QI(NN),MS,MP,ARE,MRE,E,CMOD
      E = CMOD(QR(1),QI(1))*MRE/(ARE+MRE)
      DO 10 I = 1,NN
      E = E*MS + CMOD(QR(I),QI(I))
   10 CONTINUE
      ERREV = E*(ARE + MRE) - MP*MRE
      RETURN
      END
```

```
START OF SEGMENT 00D
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0000:0
C  00D:0004:4
C  00D:0006:0
C  00D:0008:5
C  00D:000E:0
C  00D:0010:5
C  00D:0011:2
SEGMENT 00D IS 001D LONG
```

```
      DOUBLE PRECISION FUNCTION CAUCHY(NN,PT,Q)
C     CAUCHY CALCULA UNA COTA INFERIOR PARA LOS MODULOS DE LOS CEROS DE  UN
C     POLINOMIO - PT ES EL MODULO DE LOS COEFICIENTES.
      DOUBLE PRECISION Q(NN),PT(NN),X,XM,F,DX,DF,
     *DABS,DEXP,DLOG
      PT(NN) = - PT(NN)
C     CALCULA UNA ESTIMACION SUPERIOR DE LA COTA.
      N = NN - 1
      X = DEXP((DLOG(-PT(NN)) - DLOG(PT(1)))/FLOAT(N))
      IF(PT(N).EQ.0.0D0) GO TO 20
C     SI ES MEJOR EL PASO DE NEWTON EN EL ORIGEN, UTILIZARLO.
      XM = -PT(NN)/PT(N)
      IF(XM.LT.X) X = XM
C     CORTA EL INTERVALO (0,X) HASTA QUE F = 0.
   20 XM = X*.1D0
      F = PT(1)
      DO 30 I = 2,NN
      F = F*XM + PT(I)
   30 CONTINUE
      IF(F.LE.0.0D0) GO TO 40
      X = XM
      GO TO 20
   40 DX = X
C     EFECTUA LA ITERACION DE NEWTON HASTA QUE X CONVERGE HASTA DOS CIFRAS
C     DECIMALES.
   50 IF(DABS(DX/X).LE..005D0) GO TO 70
      Q(1) = PT(1)
      DO 60 I = 2,NN
      Q(I) = Q(I-1)*X + PT(I)
   60 CONTINUE
      F = Q(NN)
      DF = Q(1)
      DO 65 I = 2,N
      DF = DF*X + Q(I)
   65 CONTINUE
      DX = F/DF
      X = X - DX
      GO TO 50
   70 CAUCHY = X
      RETURN
      END
```

```
START OF SEGMENT 00E
C  00E:0000:0
C  00E:0000:0
C  00E:0000:0
C  00E:0000:0
C  00E:0000:0
C  00E:0000:0
C  00E:0002:2
C  00E:0002:2
C  00E:0003:4
C  00E:0009:3
C  00E:000C:1
C  00E:000C:1
C  00E:0010:2
C  00E:0012:1
C  00E:0012:1
C  00E:0017:0
C  00E:0018:2
C  00E:001A:0
C  00E:001D:1
C  00E:001F:2
C  00E:0020:4
C  00E:0021:3
C  00E:0022:0
C  00E:0022:5
C  00E:0022:5
C  00E:0022:5
C  00E:0029:0
C  00E:002A:5
C  00E:002C:0
C  00E:0032:0
C  00E:0034:1
C  00E:0036:2
C  00E:0037:4
C  00E:0039:0
C  00E:003C:1
C  00E:003E:2
C  00E:003F:4
C  00E:0041:0
C  00E:0041:3
C  00E:0042:2
C  00E:0042:5
SEGMENT 00E IS 004D LONG
```

```
      DOUBLE PRECISION FUNCTION SCALE(NN,PT,ETA,IFIN,SMALNO,BASE)
C     REGRESA UN FACTOR ESCALA PARA MULTIPLICAR LOS COEFICIENTES DEL
C     POLINOMIO. LA ESCALA ES HECHA PARA EVITAR EL OVERFLOW Y PARA EVITA R
C     UNDERFLOW NO-DETECTADO INTERFIRIENDO EN EL CRITERIO
C     DE CONVERGENCIA. EL FACTOR ES UNA POTENCIA DE LA BASE.
C     PT = MODULO DE LOS COFFICIENTES DE P.
C     ETA,IFIN,SMALNO,BASE = CONSTANTES DESCRIBIENDO LA
C     ARITHETICA DE PUNTO FLOTANTE.
      DOUBLE PRECISION PT(NP),ETA,IFIN,SMALNO,BASE,HI,LO,
     *MAX,MIN,X,SC,DSQRT,DLOG
C     ENCUENTRA LOS MODULOS MAYOR Y MENOR DE LOS COEFICIENTES.
      HI = DSQRT(IFIN)
      LO = SMALNO/ETA
      MAX = 0.0D0
      MIN = IFIN
      DO 10 I = 1,NN
      X = PT(I)
      IF(X.GT.MAX) MAX = X
      IF(X.NE.0.0D0.AND.X.LT.MIN) MIN = X
   10 CONTINUE
C     ESCALA SOLO SI EXISTEN COMPONENTES MUY GRANDES O MUY PEQUENAS.
      SCALE = 1.0D0
      IF(MIN.GE.LO.AND.MAX.LE.HI) RETURN
      X = LO/MIN
      IF(X.GT.1.0D0) GO TO 20
      SC = 1.0D0/(DSQRT(MAX)*DSQRT(MIN))
      GO TO 30
   20 SC = X
      IF(IFIN/SC.GT.MAX) SC = 1.0D0
   30 L = DLOG(SC)/DLOG(BASE) + .5D0
      SCALE = BASE**L
      RETURN
      END
```

```
START OF SEGMENT 011
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0000:0
C  011:0001:3
C  011:0002:5
C  011:0003:4
C  011:0004:3
C  011:0006:0
C  011:000B:1
C  011:000A:0
C  011:000D:1
C  011:000F:2
C  011:000F:2
C  011:0010:2
C  011:0013:1
C  011:0014:3
C  011:0015:4
C  011:0019:0
C  011:0019:3
C  011:001A:2
C  011:001D:1
C  011:0022:0
C  011:0024:2
C  011:0024:5
SEGMENT 011 IS 0037 LONG
```

```
      SUBROUTINE CDIVID(AR,AI,BR,BI,CR,CI)
C     DIVISION COMPLEJA C = A/B, EVITANDO EL OVERFLOW.
      DOUBLE PRECISION AR,AI,BR,BI,CR,CI,R,D,T,IFIN,DABS
      IF(BR.NE.0.0D0.OR.BI.NE.0.0D0) GO TO 10
C     DIVISION POR CERO, C = INFINITY.
      CALL HCON(T,IFIN,T,T)
      CR = IFIN
      CI = IFIN
      RETURN
   10 IF(DABS(BR).GE.DABS(BI)) GO TO 20
      R = BR/BI
      D = BI + R*BR
      CR = (AR*R + AI)/D
      CI = (AI*R - AR)/D
      RETURN
   20 R = BI/BR
      D = BR + R*BI
      CR = (AR + AI*R)/D
      CI = (AI - AR*R)/D
      RETURN
      END
```

```
 START OF SEGMENT 013
C  013:00001:0
C  013:00001:0
C  013:00001:0
C  013:00001:0
C  013:00021:2
C  013:00021:2
C  013:00051:0
C  013:00051:5
C  013:00061:4
C  013:00071:1
C  013:00091:1
C  013:000A1:3
C  013:000C1:2
C  013:000E1:4
C  013:00111:0
C  013:00111:3
C  013:00121:5
C  013:00141:4
C  013:00171:0
C  013:00191:2
C  013:00191:5
SEGMENT 013 IS 002F LONG
```

```
      DOUBLE PRECISION FUNCTION CMOD(R,I)
C     MODULO DE UN NUMERO COMPLEJO EVITANDO EL OVERFLOW.
      DOUBLE PRECISION R,I,PR,AI,DABS,DSQRT
      AR = DABS(R)
      AI = DABS(I)
      IF(AR.GE.AI) GO TO 10
      CMOD = AI*DSQRT(1.0D0 + (AR/AI)**2)
      RETURN
   10 IF(AR.LE.AI) GO TO 20
      CMOD = AR*DSQRT(1.0D0 + (AI/AR)**2)
      RETURN
   20 CMOD = AR*DSQRT(2.0D0)
      RETURN
      END
```

```
 START OF SEGMENT 014
C  014:0000:0
C  014:0000:0
C  014:0000:0
C  014:0000:0
C  014:0001:1
C  014:0002:2
C  014:0003:4
C  014:0008:2
C  014:0008:5
C  014:000A:1
C  014:000E:5
C  014:000F:2
C  014:0011:4
C  014:0012:1
SEGMENT 014 IS 001F LONG
```

```
      SUBROUTINE MCON(ETA,INFINY,SMALNO,BASE)
C     MCON PROVEE DE CONSTANTES DE MAQUINA UTILIZADAS EN VARIAS PARTES D EL
C     PROGRAMA. EL USUARIO PUEDE COLOCARLAS DIRECTAMENTE O UTILIZAR LAS
C     PROPOSICIONES DE ABAJO PARA CALCULARLAS. EL SIGNIFICADO DE LAS CUA TRO
C --- CONSTANTES ES ---------------------------------------------------
C     ETA = EL MAXIMO ERROR RELATIVO DE REPRESENTACION
C     QUE PUEDE SER DESCRITO COMO EL NUMERO POSITIVO DE PUNTO-
C     FLOTANTE MAS PEQUENO TAL QUE 1.0D0 + ETA ES
C     MAYOR QUE 1.0D0.
C     INFINITY   EL NUMERO DE PUNTO FLOTANTE MAS GRANDE.
C     SMALNO  EL NUMERO POSITIVO MAS PEQUENO DE PUNTO FLOTANTE.
C     BASE   LA BASE DEL SISTEMA NUMERICO DE PUNTO FLOTANTE UTILIZADO.
C     SEA T EL NUMERO DE DIGITOS-BASE EN CADA NUMERO DE PUNTO-
C     FLOTANTE (DOBLE PRECISION). ENTONCES ETA ES 0.5*B**(1-T)
C     O B**(1-T) DEPENDIENDO DE SI SE UTILIZA REDONDEO O TRUNCAMIENTO.
C     SEA H EL EXPONENTE MAS GRANDE Y N EL EXPONENTE MAS PEQUENO
C     EN EL SISTEMA NUMERICO, ENTONCES INFINITY ES (1-BASE**(-T))*BASE** H
C     Y SMALNO ES BASE**N.
C     LOS VALORES PARA BASE,T,H,N DE ABAJO CORRESPONDEN A LA BORROUGHS 6  700.
      DOUBLE PRECISION ETA,INFINY,SMALNO,BASE
      INTEGER H,N,T
      BASE =  8.0D0
      T = 23
      H = 68
      N = -67
      ETA = BASE**(1-T)
      INFINY = BASE*(1.0D0-BASE**(-T))*BASE**(H-1)
      SMALNO = (BASE**(N+3))/BASE**3
      RETURN
      END
```

```
 START OF SEGMENT 015
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0000:0
C  015:0001:1
C  015:0002:0
C  015:0002:5
C  015:0003:5
C  015:0006:3
C  015:000C:3
C  015:0011:2
C  015:0011:5
SEGMENT 015 IS 0025 LONG
```

```
      SUBROUTINE POLORI(ZEROR,ZEROI,OPR,OPI,XR,XI,YR,N)
C     PRUEBA DE LA EXACTITUD DE NUESTRA SOLUCION; RECONSTRUCCION DEL PO-
C     LINOMIO ORIGINAL A PARTIR DE LAS RAICES ENCONTRADAS Y DE SU PRIMER
C     COEFICIENTE.
      DOUBLE PRECISION ZEROR(49),ZEROI(49),OPR,OPI,XR(100),XI(100),
     *YR(100),XRO,XIO
      XRO = OPR
      XIO = OPI
      N1 = N + 1
      XR(N1) = -ZEROR(1)
      XI(N1) = -ZEROI(1)
      XR(N) = 1.0
      XI(N) = 0.0
      DO 10 I = 2, N-1
      XR(I) = 0.0
      XI(I) = 0.0
   10 CONTINUE
      DO 30 J = 2,N
      DO 20 I = 2,N
      YR(I) = XR(I)
      XR(I) = -(ZEROR(J)*XR(I) - ZEROI(J)*XI(I)) + XR(I+1)
      XI(I) = -(ZEROR(J)*XI(I) + ZEROI(J)*YR(I)) + XI(I+1)
   20 CONTINUE
      YR(N1) = XR(N1)
      XR(N1) = -(ZEROR(J)*XR(N1) - ZEROI(J)*XI(N1))
      XI(N1) = -(ZEROR(J)*XI(N1) + ZEROI(J)*YR(N1))
   30 CONTINUE
      XR(1) = 1.0
      XI(1) = 0.0
      DO 40 I = 1,N1
      YR(I) = XR(I)
      XR(I) = XR(I)*XRO - XI(I)*XIO
      XI(I) = YR(I)*XIO + XI(I)*XRO
   40 CONTINUE
      RETURN
      END
```

```
START OF SEGMENT 016
C  016:0000:0
C  016:0000:0
C  016:0000:0
C  016:0000:0
C  016:0000:0
C  016:0000:0
C  016:0000:0
C  016:0000:5
C  016:0001:4
C  016:0003:0
C  016:0005:5
C  016:0008:4
C  016:000A:5
C  016:000D:0
C  016:000E:0
C  016:0010:1
C  016:0012:2
C  016:0014:5
C  016:0016:0
C  016:0017:0
C  016:001A:3
C  016:0023:5
C  016:002D:1
C  016:002F:2
C  016:0032:5
C  016:003A:4
C  016:0042:3
C  016:0044:4
C  016:0046:0
C  016:0047:2
C  016:0048:0
C  016:004B:3
C  016:0050:3
C  016:0055:3
C  016:0057:4
C  016:0058:1
SEGMENT 016 IS 0060 LONG
```

ENTRADA

N = 9

COEFICIENTES

1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
0.0000000000000000D 00000	0.0000000000000000D 00000
0.0000000000000000D 00000	0.0000000000000000D 00000

SALIDA

RAICES

0.0000000000000000D 00000	0.0000000000000000D 00000
0.0000000000000000D 00000	0.0000000000000000D 00000
7.0710678118654752D-00001	7.0710678118654752D-00001
-7.0710678118654752D-00001	7.0710678118654752D-00001
-7.0710678118654752D-00001	-7.0710678118654752D-00001
7.0710678118654752D-00001	-7.0710678118654752D-00001
1.4508631998760111D-00022	1.0000000000000000D 00000
-1.0000000000000000D 00000	9.2150575336907158D-00023
-6.6174449004242214D-00022	-1.0000000000000000D 00000

POLINOMIO RECONSTRUIDO

1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	0.0000000000000000D 00000
1.0000000000000000D 00000	1.9852334701272664D-00023
1.0000000000000000D 00000	-3.2096103411357401D-00023
1.0000000000000000D 00000	1.5054687148465104D-00021
1.0000000000000000D 00000	-2.1175823681357508D-00022
1.0000000000000000D 00000	-1.2440796412797536D-00021
1.0000000000000000D 00000	-2.6039795247599304D-00022
0.0000000000000000D 00000	0.0000000000000000D 00000
0.0000000000000000D 00000	0.0000000000000000D 00000

TIEMPO DE EJECUCION = 1.38SEGS, TIEMPO DE ENTRADA Y SALIDA = 0.91SEGS.

A P E N D I C E

Formulación del Algoritmo en el Caso General

APENDICE: FORMULACION DEL ALGORITMO EN EL CASO GENERAL

A.1 El Caso de Raíces Múltiples. En la primera sección de este capítulo establecimos un algoritmo para calcular las raíces del polinomio con coeficientes complejos

$$P(z) = z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n = \prod_{j=1}^{r} (z - \rho_j)^{m_j}, \qquad (A.1)$$

con $m_1 + m_2 + \ldots + m_r = n$, habiendo supuesto durante la motivación que las raíces ρ_j eran simples y de módulos posiblemente iguales. Pasemos ahora a analizar las razones por las que dicho algoritmo funciona en general, es decir, cuando

$$|\rho_1| \leq |\rho_2| \leq \ldots \leq |\rho_r|, \quad r < n. \qquad (A.2)$$

En el caso de raíces simples utilizamos la fórmula

$$H(\lambda, s_\lambda, z) = \sum_{i=1}^{n} c_i(\lambda, s_\lambda) \frac{P(z)}{(z - \rho_i)\, P'(\rho_i)}$$

para definir a los polinomios $H(\lambda, s_\lambda, z)$; cuando existen raíces múltiples la razón por la cual no se puede aplicar esta definición es que el factor $P'(\rho_i)$ del denominador se anula. En lo que sigue veremos que éste no es un problema grave y que dicho factor es sólo un caso particular del caso general $P^{(mi)}(\rho_i)/m_i$ que no presenta dificultades de este tipo, estando definidos los correspondientes polinomios $P_i(z)$ por la fórmula

$$P_i(z) = \frac{m_i\, P(z)}{P^{(mi)}(\rho_i)\ (z - \rho_i)} \ . \qquad (A.3)$$

Veamos antes algunas relaciones que utilizaremos.

En virtud de (A.1), se puede escribir

$$P(z) = (z - \rho_i)^{mi}\ R_i(z) \ , \qquad (A.4)$$

donde

$$R_i(z) = \prod_{j \neq i} (z - \rho_j)^{mj} \quad ;$$

de aquí,

$$P^{(N)}(z) = \sum_{k=0}^{n} \binom{N}{k} (N - k)\,! \left(\frac{m_i}{m_i - (N - k)} \right) (z - \rho_i)^{mi-(N-k)}\ R_i^{(k)}(z) \ ,$$

$0 \leqslant N \leqslant m_i$, por lo tanto

$$P^{(mi)}(z) = \sum_{k=0}^{mj} \binom{m_i}{k}^2 (m_i - k)\,!\ (z - \rho_i)^k\ R_i^{(k)}(z) \quad ,$$

siendo evidente que

$$P^{(m_i)}(\rho_i) = m_i ! \; R_i(\rho_i) ,$$

$$P^{(N)}(\rho_i) = 0, \quad 0 \leq N < m_i . \tag{A.5}$$

Recuérdese ahora que los factores $P'(\rho_i)$ en el caso de raíces simples, servían para normalizar a los polinomios

$$\frac{P(z)}{z - \rho_i} , \tag{A.6}$$

ya que

$$\frac{P(\rho_j)}{(\rho_j - \rho_i)} = \delta_{ij} \; P'(\rho_i), \; i, j = 1, 2, \ldots, n. \tag{A.7}$$

Por tanto, lo que va a cambiar en el caso de raíces múltiples no es otra cosa que las constantes de normalización pues como podrá verse de (A.5)

$$\left(\frac{P(z)}{z - \rho_i}\right)^{(m_i-1)}_{z=\rho_i} = (m_i - 1) ! \; R_i(\rho_i) = (m_i - 1) ! \; \frac{P^{(m_i)}(\rho_i)}{m_i !} = \frac{P^{(m_i)}(\rho_i)}{m_i} ,$$

$$\left(\frac{P(z)}{z - \rho_i}\right)^{(N)}_{z=\rho_i} = 0, \quad 0 \leq N < m_i - 1 , \tag{A.8}$$

$$\left(\frac{P(z)}{z - \rho_j}\right)^{(N)}_{z=\rho_i} = 0, \quad 0 \leq N < m_i, \; j \neq i ,$$

y de aquí, por la manera en que los definimos, los polinomios $P_i(z)$ resultan normalizados, i.e.,

$$P_i^{(m_i-1)}(\rho_i) = 1\,,$$

$$P_i^{(N)}(\rho_i) = 0, \quad 0 \leqslant N < m_i - 1\,, \tag{A.9}$$

$$P_j^{(N)}(\rho_i) = 0, \quad 0 \leqslant N < m_i, \quad j \neq i\,.$$

Esto significa que si queremos que $H(\lambda, s_\lambda, z)$ sea una combinación lineal de los polinomios $P_i(z)$, esto es,

$$H(\lambda, s_\lambda, z) = \sum_{i=1}^{r} c_i(\lambda, s_\lambda)\, P_i(z)\,, \tag{A.10}$$

tendrá entonces las siguientes propiedades

$$H^{(m_i-1)}(\lambda, s_\lambda, \rho_i) = c_i(\lambda, s_\lambda)\,,$$

$$H^{(N)}(\lambda, s_\lambda, \rho_i) = 0, \quad 0 \leqslant N < m_i - 1\,, \tag{A.11}$$

y como queremos que $c_i(\lambda, s_\lambda) \neq 0$ para toda λ, esto quiere decir que $H(\lambda, s_\lambda, z)$ tiene las mismas raíces que $P(z)$ sólo que de multiplicidades $m_i - 1$; entonces el polinomio $H(0, 0, z)$ inicial que demos debe satisfacer esta propiedad. Construir polinomios que cumplan esto es un problema serio, pero afortunadamente contamos con uno que tiene esta propiedad y que se relaciona con $P(z)$ de manera natural, éste es el polinomio derivada; así pues,

$$H(0, 0, z) = P'(z) \tag{A.12}$$

y

$$H^{(m_i-1)}(0, 0, \rho_i) = c_i(0, 0) = c_i(0) = P^{(m_i)}(\rho_i). \tag{A.13}$$

A.2 $H(\lambda, s_\lambda, z)$ es un Polinomio de Grado $n - 1$. Consideremos primeramente el caso de un desplazamiento fijo, es decir, $s_\lambda = s$ para toda λ. Como

$$H(\lambda, s, z) = \sum_{i=1}^{r} c_i(\lambda, s) P_i(z) ,$$

este polinomio no podrá ser de grado mayor que $n - 1$ pues los $P_i(z)$ son polinomios de grado exactamente $n - 1$. Por otra parte, el coeficiente de la potencia mayor de z nos viene dado por

$$m(\lambda, s) = \sum_{i=1}^{r} \frac{c_i(\lambda, s)\, m_i}{p^{(m_i)}(\rho_i)} ,$$

de tal forma que si queremos probar que $H(\lambda, s, z)$ es un polinomio de grado $n - 1$, tendremos que demostrar que $m(\lambda, s) \neq 0$. Esto sucede para λ suficientemente grande y la demostración es esencialmente la misma que se dio en la primera sección para el caso de raíces simples. Debido a ello la omitimos y pasamos directamente a probar el mismo resultado pero para el caso de un desplazamiento s_λ variable.

Ahora

$$H(\lambda, s_\lambda, z) = \sum_{i=1}^{r} c_i(\lambda, s_\lambda) P_i(z) ,$$

y

$$m(\lambda, s_\lambda) = \sum_{i=1}^{r} \frac{c_i(\lambda, s_\lambda)\, m_i}{p^{(m_i)}(\rho_i)} . \qquad \text{(A.14)}$$

Pero para nuestros propósitos actuales nos conviene tomar

$$H(\lambda, s_\lambda, z) = \sum_{i=1}^{r} \tilde{c}_i(\lambda, s_\lambda) \cdot \frac{P(z)}{z - \rho_i} ,$$

$$m(\lambda, s_\lambda) = \sum_{i=1}^{r} \tilde{c}_i(\lambda, s_\lambda) , \qquad (A.15)$$

donde

$$\tilde{c}_i(\lambda, s_\lambda) = \frac{c_i(\lambda, s_\lambda)\, m_i}{p^{(m_i)}(\rho_i)} \qquad (A.16)$$

(es decir, lo que se ha hecho en realidad es un cambio de notación).

De manera análoga que para el caso anterior, supondremos para este caso

$$\frac{|\rho_j - s_\lambda|}{|\rho_i - s_\lambda|} < a < 1 , \qquad (A.17)$$

para toda $i \neq j$ y para toda λ. Entonces, como

$$\tilde{c}_i(\lambda, s_\lambda) = \tilde{c}_i(\lambda - 1, s_{\lambda-1})\,(\rho_i - s_\lambda)^{-1} ,$$

$$\tilde{c}_i(0, s_o) = \frac{c_i(0, s_o)\, m_i}{p^{(m_i)}(\rho_i)} = m_i ,$$

tendremos que

$$\tilde{c}_i(\lambda, s_\lambda) = m_i \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1} .$$

Sustituyendo esta expresión en (A.15)

$$m(\lambda, s_\lambda) = \sum_{i=1}^{r} m_i \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}$$

$$= m_j \prod_{k=1}^{\lambda} (\rho_j - s_k)^{-1} \left[1 + \sum_{i \neq j} \frac{m_i}{m_j} \prod_{k=1}^{\lambda} \left(\frac{\rho_j - s_k}{\rho_i - s_k}\right)\right] ,$$

de donde, por (A.17),

$$R_j(\lambda, s_\lambda) = \sum_{i \neq j} \left| \frac{m_i}{m_j} \right| \prod_{k=1}^{\lambda} \left| \frac{\rho_j - s_k}{\rho_i - s_k} \right| < a^\lambda \sum_{i \neq j} \left| \frac{m_i}{m_j} \right| .$$

Por lo tanto, existirá un natural $\lambda_o\ (N_o)$ tal que

$$R_j(\lambda, s_\lambda) < \frac{1}{N_o} \quad , \quad N_o \geq 2 \ ,$$

para toda $\lambda \geq \lambda_o\ (N_o)$, y de aquí

$$m(\lambda, s_\lambda) \neq 0, \quad \lambda \geq \lambda_o \ ,$$

con lo que se prueba que los polinomios de la sucesión $[H(\lambda, s_\lambda, z)]$ son de grado $n - 1$.

A.3 $\tilde{H}(\lambda, s_\lambda, z) \xrightarrow[\lambda \to \infty]{} P(z)/(z - \rho_j)$. Considerando a los polinomios mónicos de grado $n - 1$

$$\tilde{H}(\lambda, s_\lambda, z) = \frac{H(\lambda, s_\lambda, z)}{m(\lambda, s_\lambda)} \quad , \qquad (A.18)$$

se consigue que el límite de la sucesión $[H(\lambda, s_\lambda, z)]$ cuando $\lambda \to \infty$ (si es que existe), sea único.

Nuevamente, la demostración de este hecho para el caso de un desplazamiento fijo vuelve a ser enteramente análoga a la dada en la primera sección de este capítulo para el caso de raíces simples; por ello, pasamos directamente a

probar lo mismo para cuando el desplazamiento es variable.

$$\tilde{H}(\lambda, s_\lambda, z) = \frac{H(\lambda, s_\lambda, z)}{m(\lambda, s_\lambda)}$$

$$= \frac{\sum_{i=1}^{r} \tilde{c}_i(\lambda, s_\lambda) \frac{P(z)}{(z-\rho_i)}}{\sum_{i=1}^{r} \tilde{c}_i(\lambda, s_\lambda)}$$

$$= \frac{\sum_{i=1}^{r} m_i \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1} \frac{P(z)}{(z-\rho_i)}}{\sum_{i=1}^{r} m_i \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}$$

$$= \frac{m_j \prod_{k=1}^{\lambda} (\rho_j - s_k)^{-1} \frac{P(z)}{(z-\rho_j)} \left[1 + \sum_{i \neq j} \frac{m_i}{m_j} \left(\frac{z-\rho_j}{z-\rho_i}\right) \prod_{k=1}^{\lambda} \left(\frac{\rho_j - s_k}{\rho_i - s_k}\right)\right]}{m_j \prod_{k=1}^{\lambda} (\rho_j - s_k)^{-1} \left[1 + \sum_{i \neq j} \frac{m_i}{m_j} \prod_{k=1}^{\lambda} \left(\frac{\rho_j - s_k}{\rho_i - s_k}\right)\right]}$$

$$= \frac{\frac{P(z)}{(z-\rho_j)} \left[1 + \sum_{i \neq j} \frac{m_i}{m_j} \left(\frac{z-\rho_j}{z-\rho_i}\right) \prod_{k=1}^{\lambda} \left(\frac{\rho_j - s_k}{\rho_i - s_k}\right)\right]}{1 + \sum_{i \neq j} \frac{m_i}{m_j} \prod_{k=1}^{\lambda} \left(\frac{\rho_j - s_k}{\rho_i - s_k}\right)},$$

en donde, tomando límites,

$$\tilde{H}(\lambda, s_\lambda, z) \xrightarrow[\lambda \to \infty]{} \frac{P(z)}{z - \rho_j} .$$

Con lo discutido en las tres primeras secciones de este Apéndice se notará que el algoritmo discutido para el caso de raíces simples no es un caso particular del que aquí se ha visto para el caso general de raíces múltiples, sino que COINCIDE EXACTAMENTE CON EL.

A.4 La Sucesión (s_λ) está Bien Definida. La demostración de que el proceso iterativo que hemos descrito está siempre bien definido, esto es,

$$\tilde{H}(\lambda + 1, s_\lambda, s_\lambda) \neq 0, \quad \lambda \geqslant L,$$

en

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\tilde{H}(\lambda + 1, s_\lambda, s_\lambda)},$$

donde L es el número de iteraciones efectuadas durante los dos primeros pasos del algoritmo, se da con toda claridad y detalle en (refs 9, 10) no teniendo caso transcribirla aquí.

A.5 El Algoritmo es Precisamente una Iteración Newton–Raphson. Pasemos a probar ahora uno de los hechos más importantes de cuantos se vieron y que corresponde al de la AFIRMACION (I) dada en la sección 3.1. Este hecho nos permite afirmar que el proceso iterativo discutido a todo lo largo de este capítulo es PRECISAMENTE una iteración Newton–Raphson para una cierta función racional. La valía real de esta afirmación quedará de manifiesto cuando en la última sección de este Apéndice digamos algo acerca de la razón de convergencia del método.

Consideremos

$$\frac{P(z)}{H(\lambda, s_\lambda, z)} = \frac{P(z)}{\sum_{i=1}^{r} m_i \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1} \dfrac{P(z)}{(z - \rho_i)}} = \frac{1}{\sum_{i=1}^{r} \dfrac{m_i}{(z - \rho_i)} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}},$$

obteniendo de aquí

$$\frac{P(s_\lambda)}{H(\lambda, s_\lambda, s_\lambda)} = \frac{-1}{\sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}, \qquad (A.19)$$

de donde, a su vez,

$$\frac{P(s_\lambda)}{H(\lambda + 1, s_\lambda, s_\lambda)} = \frac{-1}{\sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)^2} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}. \qquad (A.20)$$

Por otro lado, derivando $P(z)/H(\lambda, s_\lambda, z)$ y valuando en $z = s_\lambda$, tendremos

$$\left[\frac{P(z)}{H(\lambda, s_\lambda, z)}\right]'_{z=s_\lambda} = \frac{\sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)^2} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}{\left(\sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}\right)^2};$$

de esta expresión y de (A.19), se tendrá entonces

$$\frac{\frac{P(s_\lambda)}{H(\lambda, s_\lambda, s_\lambda)}}{\left[\frac{P(z)}{H(\lambda, s_\lambda, z)}\right]'_{z=s_\lambda}} = - \frac{\sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}{\sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)^2} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}. \qquad (A.21)$$

Pero de (A.20) y de

$$m(\lambda + 1) = \sum_{i=1}^{r} \frac{m_i}{(\rho_i - s_\lambda)} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1},$$

obtenemos

$$\frac{P(s_\lambda)}{\tilde{H}(\lambda+1,\, s_\lambda,\, s_\lambda)} = \frac{P(s_\lambda)}{\dfrac{H(\lambda+1,\, s_\lambda,\, s_\lambda)}{m(\lambda+1)}}$$

$$= -\frac{\sum_{i=1}^{r} \dfrac{m_i}{(\rho_i - s_\lambda)} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}}{\sum_{i=1}^{r} \dfrac{m_i}{(\rho_i - s_\lambda)^2} \prod_{k=1}^{\lambda} (\rho_i - s_k)^{-1}},$$

y de aquí, por (A.21),

$$\frac{P(s_\lambda)}{\tilde{H}(\lambda+1,\, s_\lambda,\, s_\lambda)} = \frac{P(s_\lambda)/H(\lambda,\, s_\lambda,\, s_\lambda)}{[\,P(z)/H(\lambda,\, s_\lambda,\, z)\,]'_{z=s_\lambda}},$$

de tal manera que la fórmula

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\tilde{H}(\lambda+1,\, s_\lambda,\, s_\lambda)}$$

es idéntica a

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)/H(\lambda,\, s_\lambda,\, s_\lambda)}{[\,P(z)/H(\lambda,\, s_\lambda,\, z)\,]'_{z=s_\lambda}},$$

es decir, no es más que el método de Newton–Raphson aplicado a la función racional $P(z)/H(\lambda,\, s_\lambda,\, z)$.

A.6 Construcción de los Polinomios H(λ, s, z). Consideraremos la construcción de los polinomios con desplazamiento fijo, i.e., con $s_\lambda = s$ para toda λ. La demostración de la expresión dada en la AFIRMACION(II) de la primera sección y que cubre el caso de un desplazamiento s_λ variable, sigue un procedimiento análogo.

¿Cómo construir los polinomios H(λ, s, z) prescindiendo del conocimiento de las raíces de P(z) y de tal manera que se sigan conservando sus propiedades?

Para ello, tratemos de obtener una ecuación en diferencias finitas para H(λ, s, z) y H(λ + 1, s, z) a partir de la cual podamos determinar una fórmula de recurrencia para la obtención de las H(λ, s, z).

Como

$$H(\lambda, s, z) = \sum_{i=1}^{r} \frac{m_i (\rho_i - s)^{-\lambda}}{(z - \rho_i)} P(z), \qquad (A.22)$$

lo primero que se nos ocurre hacer es dividir tanto H(λ, s, z) como H(λ + 1, s, z) por P(z) para eliminar a este factor de las correspondientes sumatorias; de esta forma, tendríamos

$$\frac{H(\lambda, s, z)}{P(z)} = \sum_{i=1}^{r} \frac{m_i (\rho_i - s)^{-\lambda}}{(z - \rho_i)}, \qquad (A.23)$$

$$\frac{H(\lambda+1, s, z)}{P(z)} = \sum_{i=1}^{r} \frac{m_i (\rho_i - s)^{-(\lambda+1)}}{(z - \rho_i)}.$$

Ahora bien, si queremos que estas sumas sean totalmente independientes de z, lo que podemos hacer –en vista de la similitud que guardan dichas expresiones y de que

$(z - \rho_i)$ sería denominador común en la suma de las mismas– es multiplicar $H(\lambda, s, z)/P(z)$ por un factor $q(z)$ y $H(\lambda + 1, s, z)/P(z)$ por un factor $r(z)$, ambos por determinarse, de tal manera que la suma de estos dos términos nos llevara a la cancelación del factor $(z - \rho_i)$ en los denominadores de las relaciones (A.23). Veamos cómo sería esto.

$$q(z)\,\frac{H(\lambda, s, z)}{P(z)} + r(z)\,\frac{H(\lambda + 1, s, z)}{P(z)} = \sum_{i=1}^{r} \frac{m_i\,(\rho_i - s)^{-(\lambda+1)}}{(z - \rho_i)}\,[q(z)(\rho_i - s) + r(z)], \qquad (A.24)$$

requiriendo nosotros que

$$q(z)\,(\rho_i - s) + r(z) = z - \rho_i\,, \qquad (A.25)$$

siendo la forma más sencilla de conseguirlo aquélla de tomar a $q(z)$ como una constante y a $r(z)$ como un factor lineal, i.e.,

$$q(\rho_i - s) + (z + a) = z - \rho_i\,,$$

de donde

$$a + q\rho_i - qs = -\rho_i\,,$$

una manera de conseguir lo cual es tomando $a = -s$, $q = -1$; así, tendríamos que

$$q(z) = -1\,,$$

$$r(z) = z - s,$$

y sustituyendo esto en (A.24)

$$(z - s)\frac{H(\lambda+1, s, z)}{P(z)} - \frac{H(\lambda, s, z)}{P(z)} = \sum_{i=1}^{r} m_i (\rho_i - s)^{-(\lambda+1)} ;$$

pero el segundo miembro de esta expresión, además de ser independiente de z, es el coeficiente $m(\lambda + 1)$ de z^{n-1} en $H(\lambda + 1, s, z)$, esto es,

$$H(\lambda + 1, s, z) = \frac{1}{z - s}[\, H(\lambda, s, z) + m(\lambda + 1)P(z)\,] ; \qquad (A.26)$$

pero

$$m(\lambda + 1) = - \sum_{i=1}^{r} \frac{m_i(\rho_i - s)^{-\lambda}}{(s - \rho_i)} = - \frac{H(\lambda, s, s)}{P(s)} ,$$

así que la expresión (A.26) se reduce a

$$H(\lambda + 1, s, z) = \frac{1}{z - s}\left[H(\lambda, s, z) - \frac{H(\lambda, s, s)}{P(s)} P(z)\right] ,$$

que evidentemente nos da un polinomio de grado $n - 1$.

A.7 Selección de una Cota Inferior, $\beta > 0$, para los Módulos de las Raíces del Polinomio P(z). Una cota inferior para los módulos de las raíces de (A.1) nos viene dada por el único cero positivo, β, del polinomio (ref 13)

$$F(z) = z^n + |a_1| z^{n-1} + \ldots + |a_{n-1}| z - |a_n| . \qquad (A.27)$$

En primer lugar notemos que, en efecto, (A.27) tiene solamente un cero positivo, pues

$$g(z) = z^n + |a_1| z^{n-1} + \ldots + |a_{n-1}| z \qquad (A.28)$$

toma valores en el intervalo $[0, +\infty)$ para $z \in [0, +\infty)$ además de ser continua y estrictamente creciente en este intervalo; por lo tanto, existirá una y sólo una $\beta \in [0, +\infty)$ tal que $g(\beta) = |a_n|$, es decir, una y sólo una $\beta > 0$ tal que $F(\beta) = 0$.

Ahora pasemos al resultado que se quiere probar. Considérese al polinomio

$$p(z) = z^n \quad P(z^{-1}) = a_n z^n + a_{n-1} z^{n-1} + \ldots + 1. \qquad (A.29)$$

De la misma manera que para el caso anterior, puede probarse que la ecuación

$$1 + |a_1| z + \ldots + |a_{n-1}| z^{n-1} - |a_n| z^n = 0 \qquad (A.30)$$

tiene una sola raíz positiva r. Pues bien, todos los ceros del polinomio p(z) se encuentran en el círculo $|z| \leq r$ ya que, de (A.29),

$$|p(z)| \geq |a_n| |z|^n - (1 + |a_1| |z| + \ldots + |a_{n-1}| |z|^{n-1}),$$

y si $|z| > r$, el miembro derecho de esta expresión es positivo pues el miembro izquierdo de (A.30) es negativo para $r < z \leq +\infty$, y por lo tanto, $p(z) \neq 0$, $|z| > r$.

Pero debido a la forma en que se escogió al polinomio p(z), i.e.,

$$p(z) = z^n \; P(z^{-1}),$$

lo que en realidad se ha probado es que todas las raíces del polinomio P(z) se encuentran en el exterior del círculo $|z| \leqslant 1/r = \beta$, esto es, que β es una cota inferior para los módulos de las raíces del polinomio P(z), como se quería demostrar.

A.8 Razón de Convergencia. Para finalizar con este Apéndice comentemos brevemente la razón de convergencia del método aquí analizado.

Hemos visto como la sucesión de funciones racionales $[\frac{P(z)}{H(\lambda, s_\lambda, z)}]$ converge al factor lineal $z - \rho_j$. Evidentemente esto sucede con una cierta razón de convergencia. Si a esto añadimos el hecho probado en anterior sección de que mediante la fórmula

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\tilde{H}(\lambda + 1, s_\lambda, s_\lambda)}$$

lo que hacemos en realidad es aplicar el método de Newton–Raphson –que como es sabido posee una razón de convergencia cuadrática (ref 3)– a dicha sucesión, se obtendrá como resultado un método poderoso de convergencia superior a la cuadrática.

El motivo de que no se pruebe nada aquí sobre este hecho es el mismo que se dio cuando afirmamos que la sucesión $[s_\lambda]$ está bien definida, es decir, al igual que este resultado el de la razón de convergencia es tratado por Jenkins y Traub con toda claridad en (refs 9, 10), no teniendo caso reproducir aquí sus deducciones.

Con esto terminamos de discutir la parte esencialmente matemática del algoritmo.

REFERENCIAS

1. Ahlfors, L. V. (1966): **Complex Analysis**, *McGraw-Hill Book Co.*, New York.

2. Dejon, B. y Henrici, P. (Editores) (1969): **Constructive Aspects of the Fundamental Theorem of Algebra**, *Wiley*, London.

3. Henrici, P. (1964): **Elements of Numerical Analysis**, *John Wiley and Sons, Inc.*, New York y London.

4. ——— (1966): **Quotient-Difference Algorithms**, *Math. Meth. Dig. Comput.*, 2:37-62.

5. ——— y Watkins, B. O. (1965: **Finding Zeros of a Polynomial by the Q–D Algorithm**, *Comm. ACM*, 8:570-574.

6. Householder, A. S. (1964): **The Theory of Matrices in Numerical Analysis**, *Blaisdell Publishing Company*, New York.

7. ——— (1970): **The Numerical Treatment of a Single Nonlinear Equation**, *McGraw-Hill Book Co.*, New York.

8. ——— (1971): **The Padé Table, the Frobenius Identities and the qd Algorithm**, *Linear Algebra and its Applications*, 4:175-182.

9. Jenkins, M. A. (1969): **Three-stage Variable-Shift Iterations for the Solution of Polynomial Equations with a posteriori Error Bounds for the Zeros.** Stanford Dissertation.

10. ——— y Traub, J. F. (1970): **A Three-stage Variable-shift Iteration for Polynomial Zeros and its Relation to Generalized Rayleigh Iteration**, *Numer. Math.*, 14:252-263.

11. Jenkins, M. A. y Traub, J. F. (1972): **Zeros of a Complex Polynomial,** *Comm. ACM,* Algorithm 419, 15:97-99.

12. Lehmer, D. H. (1961): **A Machine Method for Solving Polynomial Equations,** *J. Assoc. Comput. Mach.,* 8: 151-162.

13. Marden, M. (1949): **The Geometry of the Zeros of a Polynomial in a Complex Variable,** *Providence Rhode Island:* Amer. Math. Soc.

14. Parlett, B. (1964): **The Development and Use of Methods of LR Type,** *SIAM Review,* 6:275-295.

15. Wall, H. S. (1948): **Analytic Theory of Continued Fractions,** *D. Van Nostrand Co., Inc.,* Princeton, N. J.

16. Wilkinson, J. H. (1959): **The Evaluation of the Zeros of Ill-conditioned Polynomials,** Num. Math., 1:150-180.

17. ——— (1963): **Rounding Errors in Algebraic Processes,** *Her Majesty's Stationery Office,* London.