BIBLIOTECA DE PROGRAMAS

Subrutina para el cálculo de las raices reales de un polinomio por el método QD.

Por F.J. Rodríguez López-Cañizares.

1- DESCRIPCION DE LA SUBRUTINA

SUBRUTINA QD

Halla las raices reales de un polinomio de grado n de coeficientes reales. No se necesita el conocimiento de ninguna aproximación previa de dichas raíces. El método empleado es el método QD (Quotient-Difference, que hace alusión a que el algoritmo consiste en la combinación de cocientes y diferencias). Esta subrutina ha sido incorporada en la cinta magnética RE-75 y puede ser usada con las siguientes tarjetas de control.

```
   1          8          16
$JOB
$* MONTAR LA CINTA RE-75 EN A6
$PAUSE
$ATTACH                 A6
$AS                     SYSLB4
$EXECUTE                IBJOB
$IBJOB
$IBFTC        nombre

                (programa principal y subrutinas del usuario)

$IEDIT                  SYSLB4, SCHF1
$IBFTC        QD1
$IEDIT
$DATA
 (datos)  si hay datos
$EOF
$IBSYS
$* DESMONTAR LA CINTA DE A6
$PAUSE
```

La llamada es la subrutina se hace de la forma:

```
CALL QD(N,M,NMAX,EPS,A,LL,NR,R,E,Q)
```

donde los parámetros tienen el siguiente significado:

N - Grado del polinomio

M - Grado del polinomio. Si para varios polinomios se quieren utilizar las mismas matrices A,EyQ, grado del polinomio de mayor grado.

NMAX - Número máximo de iteraciones más N.
EPS - Se describe más adelante.
A - Matriz unidimensional de dimensión M+1 en el programa principal, donde se almacenan los coeficientes del polinomio en la forma.

$$P(x)= A(1)x^{N}+A(2)x^{N-1}+.....+A(N+1)$$

LL - Se describe más adelante.
NR - Número de raíces reales calculadas. Si NR=0 no hay ninguna -- raiz real.
R - Matriz unidimensional de dimensión M+1 cuyas NR primeros componentes son las NR raíces reales.
E - Matriz bidimensional que ha de ser dimensionada E(M,NMAX-1) en el programa principal
Q - Matriz bidimensional que ha de ser dimensionada Q(M,NMAX-1) en el programa principal.

N,M,NMAX, EPS y A han de ser suministrados por el usuario. LL,NR ,R,E y Q son calculados en la subrutina.

La matriz E contiene los elementos $e_k^{(n)}$ escritos más adelante - de la forma:

$$E(k,n+N) = e_k^{(n)},\ k=1,2,3,....,N-1,\ n=0,1,2,....,NMAX-1$$

La matriz Q contiene los elementos $q_k^{(n)}$ descritos más adelante de la forma:

$$Q(k,n+N) = q_k^{(n)},\ k=1,2,3,....,N,\ n=0,1,...,\ NMAX-1$$

EPS es una cota de error que sirve para parar el proceso de cálculo - cuando n=j siempre que:

$$\left|Q(k,i) - Q(k,i-1)\right| \leqslant EPS$$

para los valores de k para los que las raíces son reales, y -- i+k=j. En ese caso el valor $j(\leqslant NMAX)$ se guarda en LL.

2-DESCRIPCION DEL ALGORITMO QD

El algoritmo QD fue propuesto por E. Stiefel en 1953 y desarrollado por H. Rutishauser en 8 trabajos publicados entre 1954 y 1956.

Para la redacción de este trabajo he utilizado la descripción de P. Henrici: "The Quotient-Difference Algorithm" publicado en la revista de la National Bureau of Standard, Applied Mathematics Series,- vol. 49 en 1958, y el libro del mismo autor "Elementos de Análisis Numérico", Editorial Trillas México, 1972

El algoritmo QD es presentado por Henrici para la resolución de tres problemas.

Si A es una matriz que tiene un único autovalor de mayor módulo λ_1 es posible calcularlo de la siguiente forma: Con dos vectores iniciales casi arbitrarios x e y se forman las constantes de Schwartz.

$$S_k = x^T A^k y, \quad k=0,1,2,\ldots \qquad (x^T \text{es el traspuesto de } x)$$

entonces:

$$\lim_{k\to\infty} \frac{s_{k+1}}{s_k} = \lambda_1$$

Problema 1 ¿Es posible calcular también a partir de la sucesión $\{s_k\}$ los restantes autovalores de A?.

Un segundo problema equivalente al primero es:

Problema 2 Dado el desarrollo de Taylor en el origen de una función f(z), que se sabe que es racional, determinar los polos de f(z).

Problema 3 Dado el desarrollo de Taylor en el origen de una función f(z) que se sabe que es analítica en $|z|<R$, determinar los polos de f(z) en $|z|<R$.

Para el estudio de estos tres problemas son básicos los determinantes de Hankel asociados con la función $f(z)=\sum_{n=0}^{\infty} a_n z^n$ definidos como:

(1)
$$H_0^{(n)}=1 \ , \quad H_k^{(n)} = \begin{vmatrix} a_n & a_{n+1} & \cdots & a_{n+k-1} \\ a_{n+1} & a_{n+2} & \cdots & a_{n+k} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n+k-1} & a_{n+k} & \cdots & a_{n+2k-2} \end{vmatrix} \qquad \begin{pmatrix} n=0,1,2,\ldots; \\ k=1,2,\ldots \end{pmatrix}$$

A partir de tres propiedades básicas de estos determinantes, enunciadas como lemas en el trabajo de Henrici, se llega al corolario:

(2) Corolario 1
$$\lim_{n\to\infty} \frac{H_m^{(n+1)}}{H_m^{(n)}} = \lambda_1^{m_1} \lambda_2^{m_2} \cdots \lambda_k^{m_k}$$
$$m=m_1+m_2+\cdots+m_k$$

donde $z_j=\frac{1}{\lambda_j}$, $j=1,2,\ldots k$ son los polos de f(z) de orden de multiplicidad $m_1,m_2,\ldots,m_k$, respectivamente

y $0<|z_1|\leq|z_2|\leq\cdots\leq|z_k|<R$

Si los polos son distintos en módulo y simples entonces (2) da la solución a los problemas 1,2 y 3 según se deduce de los lemas que he omitido.

La relación (2) es cierta para m=1,2...., entonces si los polos son simples y distintos en módulo (2) da los productos de los m mayores números λ_i, con lo que se pueden calcular estos. Pero desde el punto - de vista numérico el cálculo de los determinantes de Hankel es bastante complicado por lo que Rutishauser propuso un método para obviar dicho cálculo.

Si suponemos que la función $f(z)= \sum_{n=0}^{\infty} a_n z^n$ es analítica en $|Z| \leqslant R$ y sus polos son $z_i = \frac{1}{\lambda_i}$, i= 1,2,...., N, estando numerados de tal forma que :

$$0 < |z_1| \leqslant |z_2| \leqslant \cdots \leqslant |z_n| < R$$

Si para un valor particular de $k < N$ se tiene:

$$(3) \qquad |z_{k-1}| < |z_k| < |z_{k+1}|$$

entonces como consecuencia del corolario 1 se tiene:

$$\lim_{n \to \infty} \frac{H_m^{(n+1)}}{H_m^{(n)}} = \lambda_1 \lambda_2 \ldots\ldots \lambda_m$$

para m=k y para m=k-1. Por tanto el "cociente" $q_k^{(n)}$ definido por:

$$(4) \qquad q_k^{(n)} = \frac{H_k^{(n+1)} \; H_{k-1}^{(n)}}{H_k^{(n)} \; H_{k-1}^{(n+1)}}$$

verifica

$$(5) \qquad \lim_{n \to \infty} q_k^{(n)} = \lambda_k$$

Si hacemos $H_0^{(n)}=1$ y $z_0 = 0$, con la hipótesis (3), (5) se verifica también para k=1. Entonces:

$$(6) \qquad q_1^{(n)} = \frac{H_1^{(n+1)}}{H_1^{(n)}} = \frac{a_{n+1}}{a_n}$$

Los cocientes $q_k^{(n)}$ forman una serie de columnas del esquema de - cálculo QD que describiré a continuación. Con este método, para el cálculo de los $q_k^{(n)}$ no se necesita calcular los determinantes de Hankel, - sino que se calculan con los nos auxiliares $e_k^{(n)}$, definidos como:

$$(7) \qquad e_0^{(n)} = 0, \quad e_k^{(n)} = \frac{H_{k+1}^{(n)} \; H_{k-1}^{(n+1)}}{H_k^{(n)} \; H_k^{(n+1)}} \quad , \quad (n=0,1....; \; k=1,2,..)$$

El algoritmo QD se basa en el siguiente teorema:

Teorema 1 Para todos los valores de los índices para los que están definidos los números que aparecen abajo, se cumplen las siguientes relaciones:

$$(8) \qquad q_{k+1}^{(n)} \cdot e_k^{(n)} = q_k^{(n+1)} \cdot e_k^{(n+1)}$$

$$(9) \qquad q_k^{(n)} + e_k^{(n)} = q_k^{(n+1)} + e_{k-1}^{(n+1)}$$

La relación (8) es una consecuencia de las definiciones (4) y (7), mientras que (9) es una consecuencia de un Lema sobre los determinantes de Hankel, que dice.

Lema: Para todos los valores positivos de k y n, se verifica:

$$\left(H_k^{(n)}\right)^2 - H_k^{(n-1)} H_k^{(n+1)} + H_{k+1}^{(n-1)} H_{k-1}^{(n+1)} = 0$$

cuya demostración puede verse en el trabajo de Henrici.

3- APLICACIONES DEL ALGORITMO QD

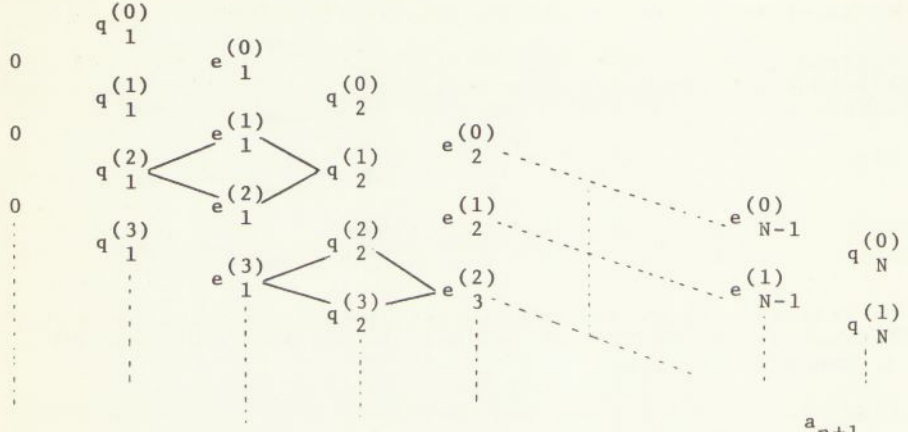
Entre las muchas aplicaciones de este algoritmo (cálculo de ceros y polos de funciones analíticas, cálculo de autovalores y autovectores de una matriz, fracciones continuas, transformación LR de Rutishauser, ceros de polinomios....) me he interesado en el cálculo de los ceros de un polinomio de coeficientes reales, distintos de cero.

Sea $P(z) = C_0 Z^N + C_1 Z^{N-1} + \ldots + C_N \quad (C_i \neq 0,\ i=0,1,\ldots,N)$

Es claro que los ceros de P(z) se pueden calcular por el algoritmo QD como los polos de la función racional.

$$(10) \qquad f(z) = \frac{1}{P(z)} = \sum_{n=0}^{\infty} a_n z^n$$

Para ello hay que construir los $q_k^{(n)}$ y $e_k^{(n)}$ utilizando las fórmulas (7) y (8) construyendo la tabla:



Para ello se construye primero la columna $q_1^{(n)} = \frac{a_{n+1}}{a_n}$ por (6) donde los a_n son los que aparecen en (10), y a continuación las columnas $e_1^{(n)}$, $q_2^{(n)}$, ... $e_{N-1}^{(n)}$ $q_N^{(n)}$ a partir de diferencias y cocientes alternativamente mediante el uso de las fórmulas (8) y (9) puestas en la forma:

$$(9') \qquad e_k^{(n)} = \left(q_k^{(n+1)} - q_k^{(n)}\right) + e_{K-1}^{(n+1)} \qquad \text{que se deduce de (9)}$$

$$(8') \qquad q_{k+1}^{(n)} = \frac{e_k^{(n+1)}}{e_k^{(n)}}\, q_k^{(n+1)} \quad , \qquad \text{que se deduce de (8)}$$

Estas fórmulas se pueden memorizar teniendo en cuenta que en la tabla anterior cada uno de los rombos que se pueden formar como los señalados en la figura, (8) quiere decir que el producto de los dos elementos del extremo superior derecho y el de los dos elementos del inferior izquierdo de un rombo centrado en una columna e, son iguales. La fórmula (9) indica que las sumas de los mismos elementos de un rombo centrado en una columna q son iguales.

Desafortunadamente el esquema de cálculo (6),(9'),(8') es inestable debido al hecho de que los números $e_k^{(n)}$ se hacen cada vez más pequeños por lo que cada columna que aparece como multiplicación de la anterior por cocientes de números muy pequeños.

La forma progresiva del algoritmo QD, obtiene los elementos de la tabla anterior por filas, es numericamente estable y se define de la forma:

Algoritmo progresivo QD Sean $C_0, C_1, \ldots\ldots, C_N$ constantes distintas de cero. Sea

$$(10) \qquad q_1^{(0)} = \frac{-C_1}{C_0}, \qquad q_{1-k}^{(k)} = 0, \qquad k=2,3, \ldots\ldots, N;$$

$$(11) \qquad e_{(k)}^{(1-k)} = \frac{C_{k+1}}{C_k}, \qquad k=1,2, \ldots\ldots, N-1$$

Considerar los elementos así generados como las dos primeras filas de un esquema QD y generar las siguientes filas mediante:

$$(12) \qquad q_k^{(n+1)} = \left(e_k^{(n)} - e_{k-1}^{(n+1)}\right) + q_{(k)}^{n}$$

$$(13) \qquad e_k^{(n+1)} = \frac{q_{k+1}^{(n)}}{q_k^{(n+1)}} e_k^{(n)}$$

y las condiciones

$$(14) \qquad e_0^{(n)} = e_N^{(n)} = 0 \quad , \qquad n=1,2, \ldots.$$

Entonces se demuestra que si el esquema generado por este algoritmo existe, es idéntico al esquema QD del polinomio:

$$P(z) = C_0 Z^N + C_1 Z^{N-1} + \ldots\ldots + C_N$$

tal como se obtuvo antes mediante (6), (9') y (8').

La forma progresiva del algoritmo ha sido la usada en la subrutina QD.

Es interesante señalar que la velocidad de convergencia de los $q_k^{(n)}$ hacia los ceros del polinomio Z_k es lineal y asintoticamente igual al mayor de los cocientes

$$\frac{Z_k}{Z_{k-1}} \qquad y \qquad \frac{Z_{k+1}}{Z_k}$$

Naturalmente si dos raíces son iguales la velocidad de convergencia es muy lenta.

En cualquier caso es recomendable no obtener la aproximación final de las raíces por este método sino utilizar el esquema para obtener unas primeras aproximaciones de ellas y luego perfeccionarlas con el método de Newton cuya convergencia es cuadrática.

Cuando en las columnas $q_k^{(n)}$ aparecen números positivos y negativos es una señal de que hay raíces complejas en cuyo caso se puede utilizar el esquema QD para generarlas siguiendo las indicaciones de Henrici en su libro en las páginas 190, 191 y 192.

Apéndice:

A continuación se dá el listado de la subrutina QD, precedido del de un programa de prueba que la utiliza para resolver la ecuación:

$$128x^4-256x^3+160x^2-32x+1 = 0$$

y escribir la tabla correspondiente al algoritmo QD progresivo, que se lista a continuación para que pueda verse la velocidad de convergencia del método.

PACO - CPN - SENTENCIA FUENTE - IFN(S) -

```
      DIMENSION A(6),E(4,34),Q(4,34),R(4)
    9 READ(5,100)N,NMAX                                                    1
      NMAXN=NMAX+N
      N11=N+1
      READ (5,101)(A(I),I=1,N11)                                           5
      WRITE(6,202)(A(I),I=1,N11)                                           12
      CALL QD(N,4,NMAXN,1.E-7,A,LL,NR,R,E,Q)                               19
      NMAX1=LL-1
      NN=LL-N
      IF(NR.EQ.0)GO TO 10
      WRITE(6,203)NN,(R(I),I=1,NR)                                         25
   10 DO 7 L=N,NMAX1
      DO 7 K=1,N,4
      J=L+1-K
      IF(J-1)1,1,2
    2 A1=Q(K+1,J-1)
      B1=E(K+1,J-1)
      IF(J-2)3,3,4
    4 A2=Q(K+2,J-2)
      B2=E(K+2,J-2)
      IF(J-3)5,5,6
    6 A3=Q(K+3,J-3)
      GOTO 8
    1 A1=0.00
      B1=0.00
    3 A2=0.00
      B2=0.00
    5 A3=0.00
    8 WRITE(6,200)Q(K,J),A1,A2,A3                                          54
    7 WRITE(6,201)E(K,J),B1,B2                                             56
      GOTO 9
  200 FORMAT(2X,4(E16.8,16X))
  201 FORMAT(2X,3(16X,E16.8))
  100 FORMAT(2I5)
  101 FORMAT(8F10.0)
  202 FORMAT(1H1,26HCOEFICIENTES DEL POLINOMIO//2X,5E16.8//)
  203 FORMAT(2X,23HRAICES CALCULADAS EN,I5,13H  ITERACIONES//2X,
     *  4E16.8//2X,8HTABLA QD)
      END
```

```
      SUBROUTINE QD(N,M,NMAX,EPS,A,L,NR,R,E,Q)
      DIMENSION          A(1),E(M,1),Q(M,1),R(1) ,IN(100)
      N1=N-1
      DO 13 I=1,N
   13 IN(I)=0
      Q(1,N)=-A(2)/A(1)
      DO 5 K=1,N1
      NK=N-K
      Q(K+1,NK)=0.DO
    5 E(K,NK+1)= A(K+2)/A(K+1)
      N2=N+2
      DO 6 L=N2,NMAX
      DO 4 K=1,N
      J=L-K-1
      IF(K.EQ.1)GOTO 1
      IF(K.EQ.N)GOTO 2
      B=E(K,J)
      C=E(K-1,J+1)
      GOTO 3
    1 B=E(K,J)
      C=0.DO
      GOTO 3
    2 B=0.DO
      C=E(K-1,J+1)
    3 Q(K,J+1)=B-C+Q(K,J)
      IF(Q(K,J+1)*Q(K,J))10,4,4
   10 IN(K)=-1
    4 CONTINUE
      DO 8 K=1,N
      IF(IN(K).EQ.-1)GOTO 9
      IF(ABS(Q(K,J+1)-Q(K,J)).GT.EPS)GOTO 9
    8 CONTINUE
        GO TO 11
    9 DO 6 K=1,N1
      J=L-K-1
    6 E(K,J+1)=(Q(K+1,J)/Q(K,J+1))*E(K,J)
      L=NMAX
   11 NR=0
      DO 12 K=1,N
      IF(IN(K).EQ.-1)GOTO 12
      NR=NR+1
      LK=L-K
      R(NR)=Q(K,LK)
   12 CONTINUE
      RETURN
      END
```

COEFICIENTES DEL POLINOMIO

0.12800000E 03 -0.25600000E 03 0.16000000E 03 -0.32000000E 02 0.10000000E 01

RAICES CALCULADAS EN 23 ITERACIONES

0.96239659E 00 0.69088483E 00 0.30865814E 00 0.38060232E-01

TABLA QD

0.20000000E 01		0.		0.	0.
	-0.62500000E 00		-0.20000000E 00	-0.31250000E-01	
0.13750000E 01		0.42500000E 00		0.16875000E 00	0.31250000E-01
	-0.19318181E 00		-0.79411763E-01	-0.57870370E-02	
0.11818182E 01		0.53877005E 00		0.24237472E 00	0.37037037E-01
	-0.88068181E-01		-0.35724710E-01	-0.88431128E-03	
0.10937500E 01		0.59111352E 00		0.27721512E 00	0.37921348E-01
	-0.47596153E-01		-0.16753854E-01	-0.12096842E-03	
0.10461538E 01		0.62195581E 00		0.29384800E 00	0.38042316E-01
	-0.28296702E-01		-0.79154925E-02	-0.15660882E-04	
0.10178571E 01		0.64233702E 00		0.30174783E 00	0.38057977E-01
	-0.17857142E-01		-0.37184260E-02	-0.19752303E-05	
0.99999997E 00		0.65647573E 00		0.30546428E 00	0.38059952E-01
	-0.11722781E-01		-0.17302183E-02	-0.24610789E-06	
0.98827719E 00		0.66646829E 00		0.30719425E 00	0.38060198E-01
	-0.79055364E-02		-0.79750700E-03	-0.30491830E-07	
0.98037165E 00		0.67357631E 00		0.30799172E 00	0.38060228E-01
	-0.54315953E-02		-0.36465884E-03	-0.37680428E-08	
0.97494005E 00		0.67864324E 00		0.30835637E 00	0.38060232E-01
	-0.37808637E-02		-0.16569071E-03	-0.46508712E-09	
0.97115918E 00		0.68225841E 00		0.30852207E 00	0.38060232E-01
	-0.26561310E-02		-0.74926505E-04	-0.57374578E-10	
0.96850305E 00		0.68483961E 00		0.30859699E 00	0.38060232E-01
	-0.18781807E-02		-0.33762787E-04	-0.70761859E-11	
0.96662486E 00		0.68668403E 00		0.30863075E 00	0.38060232E-01
	-0.13342473E-02		-0.15174715E-04	-0.87263266E-12	
0.96529061E 00		0.68800309E 00		0.30864593E 00	0.38060232E-01
	-0.95097401E-03		-0.68075476E-05	-0.10760745E-12	
0.96439964E 00		0.68894725E 00		0.30865273E 00	0.38060232E-01
	-0.67939852E-03		-0.30498244E-05	-0.13269166E-13	
0.96366023E 00		0.68962359E 00		0.30865578E 00	0.38060232E-01
	-0.48619755E-03		-0.13650141E-05	-0.16362160E-14	
0.96317403E 00		0.69010843E 00		0.30865714E 00	0.38060232E-01
	-0.34835763E-03		-0.61051470E-06	-0.20176031E-15	
0.96282567E 00		0.69045617E 00		0.30865775E 00	0.38060232E-01
	-0.24981228E-03		-0.27292115E-06	-0.24878831E-16	
0.96257585E 00		0.69070570E 00		0.30865802E 00	0.38060232E-01
	-0.17925524E-03		-0.12196121E-06	-0.30677772E-17	
0.96239659E 00		0.69088483E 00		0.30865814E 00	0.38060232E-01
	-0.12865367E-03		-0.54487113E-07	-0.37828359E-18	