"MOMENTO" Revista Depto. Física Universidad Nacional de Colombia,Número 7, Octubre 1992.

CALCULO DE RAICES COMPLEJAS DE FUNCIONES ANALITICAS

HERNAN ESTRADA
DEPARTAMENTO DE FISICA
UNIVERSIDAD NACIONAL
SANTAFE DE BOGOTA, COLOMBIA

RESUMEN

Se indica un algoritmo basado en el método de Müller para el cálculo de al menos un cero de una función compleja analítica no necesariamente polinomial.

ABSTRACT

Using the Müller method, it is shown an algorithm to calculate at least one zero of an arbitrary analytic complex function.

INTRODUCCION

El concepto de los números complejos representa una abstracción matemática que tiene particular importancia en la física, por ejemplo el cálculo de los polos de la matriz de dispersión, que son números complejos, en la teoría de colisiones nos permite obtener de manera unificada los estados acotados, estados de resonancia y estados virtuales de un sistema físico. A menudo en estos y otros problemas

nos encontramos con la necesidad de hallar ceros complejos de funciones complejas. En la literatura de métodos numéricos [1-4] se da una gran importancia al cálculo de ceros de funciones reales y poco se discute los procedimientos algorítmicos para la determinación de ceros complejos de funciones complejas (en este trabajo consideraremos solo funciones complejas analíticas que son aquellas que tanto la función como todas sus derivadas existen en la región de interés).

En las bibliotecas de computo diseñadas para grandes computadores como la **IMSL** y **NAG** se encuentran subrutinas ejecutables eficientes para este fin, pero para los sistemas personales no se dispone en las bibliotecas numéricas de las subrutinas para estos cálculos [5,6]. En algunas bibliotecas matemáticas como la suministrada por la **HEWLETT-PACKARD** sólo aparece un procedimiento para hallar las raices complejas de polinomios complejos. Aunque ésto es muy útil por ejemplo para el cálculo de los valores propios complejos de la ecuación caracteristica de una matriz, esta rutina no es suficiente para la variedad de problemas que aparecen en la física que requieren la determinación de ceros de funciones analíticas no necesariamente polinomiales.

Con el fín de remediar esta dificultad, se presenta en este artículo un algoritmo basado en el método de Müller [7] con su programa de computo escrito en **FORTRAN77** para hallar al menos un cero complejo de una función analítica.

En la próxima sección se indica el método empleado y el chequeo del programa para una función sencilla. En el apéndice se lista el código fuente empleado.

DESCRIPCION DEL ALGORITMO

En los manuales de las bibliotecas numéricas especializadas (**IMSL, NAG**) y los long write-up de las mismas se menciona el método de Müller [7] como el más adecuado para el computo de los ceros complejos.

Básicamente el método de Müller para hallar un cero aproximado de **F(Z)=0** consiste en lo siguiente: dadas tres aproximaciones p_{n-3}, p_{n-2}, p_{n-1} a la raíz exacta **p**, se construye un polinomio de interpolación cuadrático P_n a **F(Z)** que pase por los puntos p_{n-3}, p_{n-2}, p_{n-1}. Si se denota a p_n como la raíz del polinomio, que suponemos esta cerca de **p**, podemos iterativamente repetir el proceso para hallar la siguiente aproximación a **p** empleando a p_{n-2}, p_{n-1} y p_n para el ajuste polinomial. La decisión de terminar el proceso se hace como es natural imponiendo una cota al valor de **F(p)**. La geometría básica del procedimiento para el caso de una función real se indica en la figura 1. Se debe observar que conceptualmente esto no es mas que el paso posterior al método de la secante [1]. Para la interpolación cuadrática se emplea el esquema de las diferencias divididas [4] en donde la primera iteración se realiza mediante el método de la secante.

figura 1

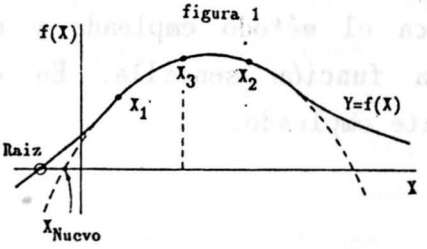
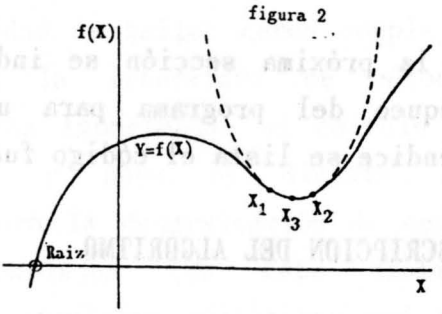


figura 2



Aunque el método de Müller da resultados bastante aceptables para las raíces, se pueden presentar situaciones fatales como la que se indica en la figura 2 en donde la busqueda falla completamente.

Teniendo en cuenta la anterior, se escribió la subrutina en doble precisión **CRAIZ** en el lenguaje **FORTRAN77** (ver apéndice) para hallar un cero de **F(Z)**. La función **F(Z)** debe ser suministrada por el usuario como la subrutina **FUNCT(Z,F)** en doble precisión. Los argumentos que se utilizan están documentados en los comentarios (ver apéndice). Los valores de entrada Z_0 y Z_1 (pueden ser reales) son los valores aproximados a la raíz y sirven como puntos de arranque.

Como ejemplo del uso de la rutina, calculamos la raíz de

$$F(Z) = \sqrt{-1} + \frac{SIN(Z)}{Z}$$

con Z_0= -2.9, Z_1= 3.7, EPSZ=1.E-8, EPSF=1.E-8

En la Tabla 1 se indica como converge el método a medida que **NZ** aumenta.

Cuando se desea encontrar otra raíz de la función **F(Z)** si es que la tiene, se debe eliminar la raíz ya calculada Z_1. Para que el programa efectivamente la busque, hacemos una nueva definición de función como **G(Z)=F(Z)/(Z-Z_1)** y aplicamos nuevamente la subrutina **CRAIZ**.

TABLA 1

iteración	Z	F(Z)
1	(-1.814690, 13.155307)	(-2074.25, 19346.09)
2	(-2.901459, -1.840161D-4)	(0.081969, 0.999933)
3	(-4.462678, -2.548335)	(-1.204680, 1.336066)
4	(-3.126902, -2.755591)	(-1.221621, -0.428331)
5	(-2.825761, -1.668135)	(-0.152728, 0.230083)
6	(-2.555258, -1.796073)	(-0.000646, 0.045321)
7	(-2.537231, -1.842768)	(-1.290210D-4, 0.001911)
8	(-2.536341, -1.844748)	(3.154786D-6, -1.136540D-6)
9	(-2.536345, -1.844748)	(-1.805947D-16, 3.330669D-16)

REFERENCIAS

[1] F. Sheid, R. S. Di Constanzo. Métodos Numéricos McGraw Hill (2^0 ed.1991)

[2] R. Bursen, J. D. Faires, A. C. Reynolds. Numerical Analysis (Second edition) PWS Publishers. Boston, Massachusetts. (1981)

[3] F. S. Acton Numerical Methods. Harper & Row Publishers (1970)

[4] D. Greenspan, V. Casulli. Numerical Analysis for Applied Matehmatics, Science and Engineering. Addison-Wesley (1988)

[5] D. E. Müller. A method for solving algebraic equations using an automatic computer. Mathematical Tables and Aids to computation 10, 208 (1956)

[6] G. Engeln-Müllges, F. Reutter. Formelsammlung für Numerischen Mathematik mit Standard FORTRAN77 Programmen.B.I. Wissenschaftsverlag (1988)

[7] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. Numerical Recipes: The art of Scientific Computing. Cambridge U. P.New York (1986).

APENDICE

```
C234567
      SUBROUTINE CRAIZ(Z0,Z1,EPSZ,EPSF,NZ,IPRINT)
C
C     RAIZ DE UNA FUNCION COMPLEJA ANALITICA F(Z)
C
C     LA FUNCION F(Z) ES SUMINISTRADA POR LA USUARIO EN LA
C     SUBROUTINE FUNCT F(Z,F) EN DONDE:
C           Z ES UN VALOR COMPLEJO EN EL QUE SE EVALUA LA
C           FUNCION (NO DEBE SER CAMBIADO POR F)
C           F ES EL VALOR CALCULADO DE LA FUNCION EN EL
C           PUNTO Z.
C     ************************************************
C     PARAMETROS DE LA SUBRUTINA:
C
C     Z0, Z1          SON LOS PUNTOS INICIALES DE PARTIDA (INPUT)
C                     (VALORES ESTIMADOS)
C     Z0, Z1          ES LA ULTIMA Y PENULTIMA SOLUCION
C                     ITERATIVA (OUTPUT)
C     EPSF            PARAMETRO DE ERROR PARA F(Z)
```

```
C       EPSZ            PARAMETRO DE ERROR PARA Z
C       NZ              NUMERO MAXIMO DE ITERACIONES
C       IPRINT.GT.0     ESCRIBE DESPUES DE CADA ITERACION
C       IPRINT.LT.0     NO IMPRIME NADA
C       ********************************************
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMPLEX*16 Z0,Z1,Z2,DZ,OM,LA,MU,CERO,CUART,MEDIO
        COMPLEX*16 F0,F1,F01,F12,F012,F123,Z3
        DATA CERO,CUART,MEDIO/(0.D0,0.D0),(.25D0,0.D0),(.5D0,0.D0)/
        F012=CERO
        CALL FUNCT(Z0,F0)
        CALL FUNCT(Z1,F1)
        F0A=CDABS(F0)
        F1A=CDABS(F1)
        F01=(F0-F1)/(Z0-Z1)
        D0=CDABS(Z0-Z1)
        DMAX=2.*D0
        IF(IPRINT.GT.0) THEN
        WRITE(6,*)'VALORES INICIALES Z0, F0 / Z1, F1 '
        WRITE(6,*) Z0,Z1
        WRITE(6,*) F0,F1
        WRITE(6,*)'---------------------------------'
        END IF
C       CICLO DE ITERACION
        DO 10 I=1,NZ
        IF(F0A.LT.F1A) GOTO 7
        Z3=Z0
        Z0=Z1
        Z1=Z3
        Z3=F0
        F0=F1
        F1=Z3
    7   Z2=Z1
        Z1=Z0
        F1=F0
        F1A=F0A
        F12=F01
        F123=F012
C       FORMULA DE ITERACION DE SEGUNDO GRADO
        OM=F12+(Z1-Z2)*F123
        LA=F1/OM
        MU=F123/OM
        IF(CDABS(LA*MU).GT.1.D0) MU=CERO
        DZ=LA/(MEDIO+CDSQRT(CUART-LA*MU))
```

```
C       EL PASO NO DEBE SER MAYOR QUE LA DIFERENCIA ENTRE LOS
C       VALORES DE ENTRADA
        D0=CDABS(DZ)
        IF(D0.GT.DMAX) DZ=DZ*(DMAX/D0)
        Z0=Z0-DZ
        IF(Z0.EQ.Z1) GOTO 21
        CALL FUNCT(Z0,F0)
        F0A=CDABS(F0)
C       DIFERENCIAS DIVIDIDAS
        F01=(F0-F1)/(Z0-Z1)
        F012=(F01-F12)/(Z0-Z2)
        IF(D0.LE.EPSZ.AND.F0A.LE.EPSF) GOTO 20
        IF(F0A.EQ.0.D0.OR.D0.EQ.0.D0) GOTO 21
        IF(IPRINT.GT.0) THEN
        WRITE(6,*)'ITERACION=',I
        WRITE(6,*)'Z, F(Z)   ',Z0,F0
        WRITE(6,*)'ABS(F(Z))=',F0A
        END IF
  10    CONTINUE
        WRITE(6,*)'   !!! NUMERO MAXIMO DE ITERACIONES !!!'
  21    IF(IPRINT.GE.0) WRITE(6,*)'   !!! CONVERGENCIA INCIERTA !!!'
  20    CONTINUE
        IF(IPRINT.GT.0) THEN
        WRITE(6,*)'VALORES FINALES'
        WRITE(6,*)'Z, F(Z)   ',Z0,F0
        WRITE(6,*)'ABS(F(Z))=',F0A
        END IF
C       ************************************************
C       ITERACION ADICIONAL, NO SE EVALUA LA FUNCION
        Z3=Z2
        Z2=Z1
        Z1=Z0
        OM=F01+(Z0-Z1)*F012
        LA=F0/OM
        MU=F012/OM
        IF(CDABS(LA*MU).GT.1.D0) MU=CERO
        Z0=Z0-LA/(MEDIO+CDSQRT(CUART-LA*MU))
        IF(IPRINT.GE.0) THEN
        WRITE(6,*)' Z0=',Z0
        WRITE(6,*)' Z1=',Z1
        WRITE(6,*)' Z2=',Z2
        WRITE(6,*)' Z3=',Z3
        END IF
        RETURN
        END
```