11.3 Conversion between Chebyshev and Monomial Representations of a Polynomial

A. Purpose

These subroutines convert a polynomial represented in the monomial basis to a representation in the Chebyshev basis, and vice versa.

B. Usage

B.1 Program Prototype, Single Precision

INTEGER N

REAL COEFF(0:≥N)

Assign values to N, and to coefficients in COEFF(). If COEFF(i) contains coefficients of $T_i(x)$, i = 0, 1, ..., N, which are to be converted to coefficients of x^i,

CALL SCONCM(N, COEFF)

For the inverse operation,

CALL SCONMC(N, COEFF)

B.1 Argument Definitions

N [in] The degree of the polynomial.

COEFF [inout] When calling SCONCM, COEFF(i) contains the coefficient of T_i, $i = 0$, 1, ... N, on input, and contains the coefficient of x^i on output. When calling SCONMC, COEFF(i) contains the coefficient of x^i, $i = 0$, 1, ..., N on input, and the coefficient of T_i on output.

B.2 Modifications for Double Precision

Change the names SCONCM and SCONMC to DCONCM and DCONMC respectively, and change the REAL declaration to DOUBLE PRECISION.

C. Examples and Remarks

The program DRSCON prints out the coefficients of the Chebyshev polynomials corresponding to x^k, k = 0, 1, ..., 6, and then prints the coefficients in the monomial basis corresponding to the Chebyshev polynomials T_k, k = 0, 1, ..., 6. Results are in the file ODSCON.

If these subroutines are applied to a coefficient array, say P(), obtained from SPFIT, Chapter 11.1, the zeroth order coefficient is in P(3) so the call would be of the form SCONxx(NDEG, P(3)), where xx is either CM or MC.

D. Functional Description

Consider the polynomial $p_n(x)$ of degree n,

$$p_n(x) = \sum_{k=0}^{n} a_k x^k \equiv \sum_{k=0}^{n} c_k T_k(x) \tag{1}$$

where $T_k(x)$ is the k^{th} Chebyshev polynomial. This software converts between the a_k's and the c_k's.

Using the well-known identities,

$$\begin{aligned} xT_k(x) &= \frac{1}{2}\left[T_{k+1}(x) + T_{k-1}(x)\right], \quad k > 1 \\ xT_0(x) &= T_1(x) = x, \end{aligned} \tag{2}$$

we can write p_n in forms intermediate between the extremes represented in Eq. (1). It is these intermediate forms that are used in obtaining the recurrences. Thus

$$p_n(x) = \sum_{k=0}^{j-1} a_k x^k + x^j \sum_{k=0}^{n-j} b_{k,j} T_k(x) \tag{3}$$

$$\equiv \sum_{k=0}^{j} a_k x^k + x^{j+1} \sum_{k=0}^{n-j-1} b_{k,j+1} T_k(x) \tag{4}$$

Note that $b_{k,0} \equiv c_k$. Using Eq. (2), Eq. (4) gives

$$p_n(x) = \sum_{k=0}^{j} a_k x^k + \frac{x^j}{2} \sum_{k=1}^{n-j-1} b_{k,j+1}\big[T_{k+1}(x) + T_{k-1}(x)\big] + x^j b_{0,j+1} T_1(x). \tag{5}$$

Collecting like terms in Eqs. (3) and (5), we obtain,

$$\begin{aligned} a_j + \frac{1}{2} b_{1,j+1} &= b_{0,j} \\ b_{0,j+1} + \frac{1}{2} b_{2,j+1} &= b_{1,j} \\ \frac{1}{2}\left[b_{k-1,j+1} + b_{k+1,j+1}\right] &= b_{k,j}, \ \ k = 2, 3, ..., n-j-2 \\ \frac{1}{2} b_{k-1,j+1} &= b_{k,j}, \ \ k \geq n-j-1. \end{aligned} \tag{6}$$

A more efficient recursion is obtained with $b_{k,j}$ replaced by $2^j B_{k,j}$. Thus,

$$\begin{aligned} 2^{-j} a_j + B_{1,j+1} &= B_{0,j} \\ 2B_{0,j+1} + B_{2,j+1} &= B_{1,j} \\ B_{k-1,j+1} + B_{k+1,j+1} &= B_{k,j}, \ \ k = 2, 3, ..., n-j-2 \\ B_{k-1,j+1} &= B_{k,j}, \ \ k \geq n-j-1. \end{aligned} \tag{7}$$

In the code, the $B_{k-j,k}$ share space with the original a_k or the original c_k. If one starts with the a_k then one runs j from n down to 0, and otherwise j runs in the opposite direction. Observe that the innermost loop requires only a single addition.

E. Error Procedures and Restrictions

If $n < 0$, a return is made without taking any action.

F. Supporting Information

The source language is ANSI Fortran 77. Algorithm and code by F. T. Krogh, JPL, January 1992.

Entry	Required Files
DCONCM	DCONCM
DCONMC	DCONMC
SCONCM	SCONCM
SCONMC	SCONMC

DRSCON

```
      program DRSCON
c>> 2001-05-22 DRSCON Krogh Minor change for making .f90 version.
c>> 1996-06-25 DRSCON Krogh  Special code for C conversion.
c>> 1994-10-19 DRSCON Krogh  Changes to use M77CON
c>> 1994-08-09 DRSCON WVS    Remove '0' from formats
c>> 1992-03-09 DRSCON Krogh  Initial Code.
c Check program for converting between Chebyshev and monomial basis.
c--S replaces "?": DR?CON, ?CONCM, ?CONMC
c
      integer NMAX
      parameter (NMAX=6)
      integer          K, N
      real             COEFF(0:NMAX)
c
c%%    printf( "         " );
c%%    for (k = 0; k <= NMAX; k++) printf( "    X**%1ld", k );
c%%    printf( "\n" );
      print '(7X, 9(:''     X**'',I1))', (K, K = 0, NMAX)
      do 20 N = 0, NMAX
         do 10 K = 0, N-1
            COEFF(K) = 0.E0
   10    continue
         COEFF(N) = 1.E0
         call SCONCM(N, COEFF)
         print '(''T'', I1, ''(X) ='', F7.3, 8F8.3)', N,
     1      (COEFF(K), K = 0, N)
   20 continue
c%%    printf( "\n       " );
c%%    for (k = 0; k <= NMAX; k++) printf( "    T%1ld(X)", k );
c%%    printf( "\n" );
      print '(/, 6X, 9(:''    T'', I1, ''(X)''))', (K, K = 0, NMAX)
      do 120 N = 0, NMAX
         do 110 K = 0, N-1
            COEFF(K) = 0.E0
  110    continue
         COEFF(N) = 1.E0
         call SCONMC(N, COEFF)
         print '(''X**'', I1, '' ='', 9F8.5)', N, (COEFF(K), K = 0, N)
  120 continue
      stop
      end
```

ODSCON

	X**0	X**1	X**2	X**3	X**4	X**5	X**6
T0(X) =	1.000						
T1(X) =	0.000	1.000					
T2(X) =	−1.000	0.000	2.000				
T3(X) =	0.000	−3.000	0.000	4.000			
T4(X) =	1.000	0.000	−8.000	0.000	8.000		
T5(X) =	0.000	5.000	0.000	−20.000	0.000	16.000	
T6(X) =	−1.000	0.000	18.000	0.000	−48.000	0.000	32.000

	T0(X)	T1(X)	T2(X)	T3(X)	T4(X)	T5(X)	T6(X)
X**0 =	1.00000						
X**1 =	0.00000	1.00000					
X**2 =	0.50000	0.00000	0.50000				
X**3 =	0.00000	0.75000	0.00000	0.25000			
X**4 =	0.37500	0.00000	0.50000	0.00000	0.12500		
X**5 =	0.00000	0.62500	0.00000	0.31250	0.00000	0.06250	
X**6 =	0.31250	0.00000	0.46875	0.00000	0.18750	0.00000	0.03125