ENUMERATION OF ELEMENTARY MULTI-INDICES FOR MULTIVARIATE FOURIER SERIES

by

JOHN H. MONAHAN

INSTITUTE OF STATISTICS MIMEO SERIES No. 1338

1 9 8 1

Enumeration of Elementary Multi-indices for Multivariate Fourier Series

Abstract

An algorithm is given that produces a set of elementatry multi-indices that facilitate the expression of multivariate Fourier series.

Key words: Multi-index, multivariate Fourier series, simulating nested loops

I. Introduction

A Fourier series expansion of a function f in $x = (x_1,\ldots,x_N)^T$ can be written as

$$(1) \qquad f(x) = \sum_{k_1} \sum_{k_2} \cdots \sum_{k_N} a_{k_1,\ldots,k_N} \exp\left\{ i \sum_{j=1}^{N} k_j x_j \right\}$$

where $i^2 = 1$. The vector $k = (k_1,\ldots,k_N)^T$ is called a *multi-index*. A typical term in the expansion (1) can then be written as

$$a_k \, e^{ik^T x} .$$

The L_1 norm is natural for the multi-index and can be written as

$$(2) \qquad \| k \| \equiv \| k \|_1 = \sum_{j=1}^{N} |k_j| .$$

Thus the finite Fourier expansion can be written analagously to (1) with the summation restricted to $\Sigma|k_j| \leq K$, a bound on the norm of the multi-index:

$$(3) \qquad f(x) \simeq \sum_{\|k\| \leq K} a_k e^{ik^T x} \equiv f_K(x)$$

The limit of the Fourier expansion is then defined by taking K arbitrarily large in the right bound side of (3).

For the Fourier series to be real-valued, the coefficients must satisfy

$$a_k = \bar{a}_{-k}$$

where the bar denotes complex conjugate. Enforcing this condition and others with economic interpretations (see Gallant, 1981) suggests a rewriting of (3) as

$$(4) \qquad f_K(x) = \sum_{\alpha=1}^{A} \sum_{j=-J}^{J} a_{j\alpha} \exp\{ijk_\alpha^T x\} + a_0 ,$$

where the vectors k_α are called *elementary multi-indices*, with α indexing these vectors. A multi-index is called *elementary* if it cannot be expressed as an integer multiple of another multi-index with a smaller norm. Since non-negative integer multiples are permitted, an additional normalization restriction must be that the first nonzero entry be positive. Hence (0,1,2), (0,1,-2) and (1,-1,-1) are elementary; (0,2,4), (0,-1,2), and (-1,1,1) are not.

The bound A in (4) represents the number of elementary multi-indices whose norm is less than K, and J, which depends on α, is the largest integer that satisfies

$$J\|k_\alpha\| \leq K .$$

The zero vector is a special case and must be treated separately.

II. The Algorithm

The problem at hand, then, is to produce a table of elementary vectors k_α such that $||k_\alpha|| \leq K$. The approach is essentially to produce all multi-indices with norm not exceeding K and deleting those that are not elementary.

Recall that the first restriction on elementary multi-indices is that it cannot be expressed as an integer multiple of another multi-index with a smaller norm. Given a prime number p, if $(k_\alpha)_j \bmod p = 0$ for all $j = 1,\ldots,N$, then k_α is not elementary and is rejected. The logical function CHECK performs this test for the first ten primes p, which should be sufficient for most practical problems. If any component is -1 or $+1$, then it cannot be a multiple and is immediately accepted.

Exploring the second normalization restriction requires some sleight of hand. This restriction forces the first nonzero component must be positive, hence the bounds on the first component are 1 and K and the others are $-K$ to $+K$. The missing multi-indices then are those that begin with zeros. These can be generated more easily in a recursive fashion. Notice that all elementary multi-indices of length N that begin with a zero can be formed by prefacing all elementary multi-indices of length $N - 1$ with a 0. So the algorithm generates elementary multi-indices with lengths 1 through N.

Since further details regarding the algorithm require lengthy discussion, the FORTRAN code in the appendix should be consulted. The features that should be noted are the simulation of nested loops, the updating for computing the norm, and the reversing of the order of the components when stored. Finally, the elementary multi-indices are sorted

by norm using a heapsort algorithm (see Knuth, 1973, pp. 145-149). The output from a test run with $K = 4$ and the length $N = 3$ is given in the appendix. Caution must be given that the number of A of K_α's can be quite large even for moderate values of N and K.

References

Gallant, A. Ronald (1981), "On the bias in flexible functional forms and and essentially unbiased form: the Fourier flexible form," J. of Econometrics 15, pp. 211-245.

Knuth, Donald E. (1973), The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley.

APPENDIX

```
00010 C  THIS PROGRAM PRODUCES ELEMENTARY MULTI-INDICES OF A GIVEN LENGTH
00020 C  WHOSE NORM IS LE TO KCAP. NORM(INDEX)=SUM OF ABS(INDEX(J)) OVER J
00030 C  NESTRE    J F MONAHAN      APRIL 1981
00040 C        DEFAULT MAX DIMENSIONS:
00050 C  MAX LENGTH OF MULTI-INDEX IS 16
00060 C  MAX NUMBER OF GOOD INDICES IS 1024
00070       INTEGER INDEX(16),IMN(16),IMX(16),NORM(17),I,J,KCAP,L,LP1,NUM,
00080      1LENGTH,P,INDICE(1024,16),STNORM(1024),INDCNT(1024)
00090       LOGICAL CHECK,OK
00100    20 FORMAT(2X,I6,' INDICES WITH NORM LE K=',I4/5X,'ALPHA',3X,
00110      1'NORM',4X,'MULTI-INDEX')
00120    21 FORMAT(4X,I4,4X,I4.4X,16I3)
00130 C  TEST VAULES
00140       KCAP=4
00150       LENGTH=3
00160 C  INITIALIZE INDEX
00170       DO 10I=1,10
00180    10 INDEX(I)=0
00190       NUM=0
00200       DO 9L=1,LENGTH
00210       LP1=L+1
00220       DO 8I=1,10
00230       IMN(I)=-KCAP
00240     8 IMX(I)=KCAP
00250       IMN(L)=1
00260       NORM(LP1)=0
00270       DO 1I=1,L
00280       J=LP1-I
00290       NORM(J)=NORM(J+1)+IABS(IMN(J))
00300 C SIMULATION OF NESTED LOOPS WITH L INDEX VARIABLES AND BOUNDS:
00310 C IMN(I)  LE  INDEX(I)   LE   IMX(I)  FOR I=1,...,L
00320    1  INDEX(I)=IMN(I)
00330    2  P=1
00340       INDEX(P)=IMN(P)
00350       NORM(P)=NORM(P+1)+IABS(INDEX(P))
00360    3  CONTINUE
00370 C THIS IS THE CENTER OF THE LOOPING
00380       IF(NORM(1).GT.KCAP) GO TO 4
00390       OK=CHECK(INDEX,KCAP,L)
00400       IF(.NOT.OK) GO TO 4
00410 C  ADDITIONAL CHECKS CAN GO IN HERE
00420 C  NOW STORE A GOOD INDEX, ITS NORM IS STORED IN STNORM
00430       NUM=NUM+1
00440       INDCNT(NUM)=NUM
00450       STNORM(NUM)=NORM(1)
00460       DO 7I=1,LENGTH
00470    7  INDICE(NUM,I)=INDEX(LENGTH+1-I)
00480 C  THIS ENDS THE CENTER LOOP
00490    4  INDEX(P)=INDEX(P)+1
00500       NORM(P)=NORM(P+1)+IABS(INDEX(P))
00510   16  IF(INDEX(P).LE.IMX(P)) GO TO (3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2),P
00520       INDEX(P)=IMN(P)
00530       P=P+1
00540       IF(P.LE.L) GO TO 4
00550    9  CONTINUE
00560 C   SORT ON NORM STORED IN STNORM, INDCNT IS CARRIED ALONG
00570       CALL HPSORT(STNORM,INDCNT,NUM)
00580       WRITE(3,20) NUM,KCAP
00590       DO 11I=1,NUM
00600       P=INDCNT(I)
00610   11  WRITE(3,21) I,STNORM(I),(INDICE(P,J),J=1,LENGTH)
00620       STOP
00630       END
```

```
00640       LOGICAL FUNCTION CHECK(INDEX,KCAP,L)
00650 C    CHECK TO SEE IF CURRENT INDEX IS AN INTEGER MULTIPLE OF ONE WITH
00660 C    SMALLER NORM.  THIS USES JUST THE FIRST 10 PRIME NUMBERS SO THAT
00670 C    IT CAN ONLY CHECK INDICES WITH NORM LESS THAN 29*29=841
00680       INTEGER INDEX(16),P,PRIME(10),J,KCAP,L
00690       DATA PRIME/2,3,5,7,11,13,17,19,23,29/
00700       CHECK=.TRUE.
00710       DO 2I=1,10
00720       P=PRIME(I)
00730       IF(P.GT.KCAP) RETURN
00740       DO 3J=1,L
00750       IF(IABS(INDEX(J)).EQ.1) RETURN
00760       IF(MOD(INDEX(J),P).NE.0) GO TO 2
00770    3  CONTINUE
00780       CHECK=.FALSE.
00790       RETURN
00800    2  CONTINUE
00810       RETURN
00820       END
00830       SUBROUTINE HPSORT(K,M,N)
00840 C   INTEGER MODIFICATION OF HEAPSORT ON K, M CARRIED ALONG
00850 C   SEE KNUTH, VOL 3, PP. 145-149, ALGORITHM H
00860       INTEGER R,M(1),MM,K(1),KK,I,J,L,N
00870       L=N/2+1
00880       R=N
00890    2  IF(L.GT.1) GO TO 1
00900       KK=K(R)
00910       K(R)=K(1)
00920       MM=M(R)
00930       M(R)=M(1)
00940       R=R-1
00950       IF(R.EQ.1) GO TO 9
00960       GO TO 3
00970    1  L=L-1
00980       KK=K(L)
00990       MM=M(L)
01000    3  J=L
01010    4  I=J
01020       J=2*J
01030       IF(J-R) 5,6,8
01040    5  IF(K(J).LT.K(J+1)) J=J+1
01050    6  IF(KK.GT.K(J)) GO TO 8
01060    7  K(I)=K(J)
01070       M(I)=M(J)
01080       GO TO 4
01090    8  K(I)=KK
01100       M(I)=MM
01110       GO TO 2
01120    9  K(1)=KK
01130       M(1)=MM
01140       RETURN
01150       END
```

```
watfiv nestre.fort
WATFIV COMPILATION
      END                                                              0001
*EXTENSION*  PSEUDO VARIABLE DIMENSIONING ASSUMED FOR ARRAY M
*EXTENSION*  PSEUDO VARIABLE DIMENSIONING ASSUMED FOR ARRAY K
COMPILATION COMPLETE
     49 INDICES WITH NORM LE K=    4
    ALPHA   NORM     MULTI-INDEX.
      1       1       0  1  0
      2       1       0  0  1
      3       1       1  0  0
      4       2       1  0  1
      5       2       1  0 -1
      6       2       0  1  1
      7       2       1 -1  0
      8       2       1  1  0
      9       2       0  1 -1
     10       3       0  1  2
     11       3       1 -2  0
     12       3       1  1 -1
     13       3       1  0  2
     14       3       0  1 -2
     15       3       0  2 -1
     16       3       0  2  1
     17       3       1 -1  1
     18       3       2  0  1
     19       3       2 -1  0
     20       3       2  1  0
     21       3       1 -1 -1
     22       3       1  2  0
     23       3       2  0 -1
     24       3       1  0 -2
     25       3       1  1  1
     26       4       1  1 -2
     27       4       1 -2 -1
     28       4       1  0  3
     29       4       1 -3  0
     30       4       0  3  1
     31       4       3  1  0
     32       4       3  0  1
     33       4       3 -1  0
     34       4       1  0 -3
     35       4       2  1  1
     36       4       1 -1  2
     37       4       2  1 -1
     38       4       2 -1  1
     39       4       2 -1 -1
     40       4       0  3 -1
     41       4       1  3  0
     42       4       1  2  1
     43       4       3  0 -1
     44       4       1 -1 -2
     45       4       1  2 -1
     46       4       0  1 -3
     47       4       0  1  3
     48       4       1 -2  1
     49       4       1  1  2
STATEMENTS EXECUTED=     6186

READY
```