EVALUATION OF THE DOMINANT LATENT ROOT AND VECTOR OF A SYMMETRIC MATRIX ON CALCULATING PUNCH TYPE IBM 626

By NANDURI CHINNAM RAJU
Indian Statistical Institute

SUMMARY. The dominant latent root and associated vector of symmetric matrices are often required in statistical and other studies. A method of such computations on the punched card system is presented in this paper. Only a sorter and calculating punch type IBM-626 or similar machine need be used.

1. Mathematical basis of the method

To evaluate the dominant latent root and the associated latent vector of a real symmetric matrix A, we adopt the method of successive approximations. First, by repetitive squaring the matrix A is raised to a high power to get

$$A^{2^m} = B = ((b_{ij})), \text{ say.}$$

Let $$x_i^{(0)} = \sum_{j=1}^{n} b_{ij}, \quad \text{and } y_i^{(0)} = x_i^{(0)}/x_1^{(0)} \; ; \quad i = 1, 2, \ldots, n.$$

Then $(y_1^{(0)}, y_2^{(0)}, \ldots, y_n^{(0)})$ can be taken as the initial approximation to the dominant latent vector of A.

The next approximation (which we shall call the first approximation) is $(y_1^{(1)}, y_2^{(1)}, \ldots, y_n^{(1)})$ where

$$y_i^{(1)} = x_i^{(1)}/x_1^{(1)}, \; x_i^{(1)} = \sum_{j=1}^{n} b_{ij} y_j^{(0)}$$

for $i = 1, 2, \ldots, n$. Successive approximations are obtained recursively from the formulae

$$y_i^{(t+1)} = x_i^{(t+1)}/x_1^{(t+1)} \; ; \quad x_i^{(t+1)} = \sum_{j=1}^{n} b_{ij} y_j^{(t)}$$

so that $[y_1^{(t+1)}, y_2^{(t+1)}, \ldots, y_n^{(t+1)}]$ forms the $(t+1)$-th approximation to the dominant latent vector. One stops when

$$y_i^{(t+1)} = y_i^{(t)} \qquad \text{for } i = 1, 2, \ldots, n$$

to the desired degree of accuracy. If y_i is the limiting value of $y_i^{(t)}$, then $(y_1, y_2, \ldots, y_n)$ is the dominant latent vector and the dominant latent root λ is obtained from the relation

$$\lambda^{2^m} = \sum_{i=1}^{n} b_{ij} y_i / y_j.$$

2. The system

One punched card for each element of the matrix B showing in addition to the value 'b_{ij}' of the element, the row number i, and the column number j is prepared. This set of n^2 cards are called the matrix cards. Now we consider the matrix

$$M^{(0)} = ((m_{ij}^{(0)})),$$

where $\quad m_{ij}^{(0)} = y_j^{(0)}, \quad$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$

and $(y_1^{(0)}, y_2^{(0)}, \ldots, y_n^{(0)})$ is the first approximation to the dominant latent vector of A. Then another set of n^2 cards, one for each element (m_{ij}) of the matrix M showing its row number i and column number j also, are prepared, which may be called multiplier cards. Also the value $x_1^{(0)} = \sum\limits_{j=1}^{n} b_{1j}$ is punched on all the cards. The same fields are allotted to show the row numbers and column numbers in both the cases. Then the multiplier cards, the matrix cards and a set of n^2 blank cards are merged in a way specified in Section 4 and are fed into the calculating punch, so that the new multiplier values $m_{ij}^{(1)}$, with the necessary row, column identifications and cycle number, are automatically punched by the machine on the set of n^2 blank cards, which may now be called new multiplier cards. In addition the value

$$x_1^{(1)} = \sum_{j=1}^{m} b_{1j} y_j^{(0)} = (b_{11}m_{11} + b_{12}m_{12} + \ldots + b_{1n}m_{1n}),$$

is also punched on all these new multiplier cards. It is to be noted that

$$m_{ij}^{(1)} = y_j^{(1)}, \quad \text{for} \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, n \quad \text{and} \quad (y_1^{(1)}, y_2^{(1)}, \ldots, y_n^{(1)})$$

becomes the 2nd approximation to the dominant latent vector of A. Then the three sets of cards are separated by sorting on the column showing the cycle number. If $x_1^{(1)} = x_1^{(0)}$, the procedure is stopped. Otherwise, all the previous operations are repeated using the new multiplier cards in place of the old multiplier cards. The whole process is stopped when $x_1^{(t+1)} = x_1^{(t)}$, and the values of $m_{ij}^{(t+1)}$ give us the limiting value of $y_j^{(t)}$.

3. Description of the method

Basic data are transferred to punched cards according to the following card designs.

CARD DESIGN FOR MATRIX CARDS

serial number	item	number of columns	card columns	remarks
1.	C.D.I.	1	1	punch 1
2.	row number	2	7–8	
3.	column number	2	9–10	
4.	value of the element (b_{ij})	5	11–15	

Note : In addition 'X' is punched in Column 7, on the card showing the element b_{1n}, and in column 9 on all the cards showing the elements $b_{i\,\overline{n-1}}$, for $i = 1, \ldots, n$.

CARD DESIGN FOR MULTIPLIER CARDS

serial number	item	number of columns	card columns	remarks
1.	C.D.I.	1	1	punch 2
2.	row number	2	7–8	
3.	column number	2	9–10	
4.	value of the element (m_{ij})	5	16–20	
5.	value of x_1	7	21–27	
6.	cycle number	1	35	

Note: The above items are punched manually for the first time, and later on they are punched automatically by the calculating punch.

Procedure: The detailed steps followed by the operators are exceedingly simple. A complete operational procedure is given below.

Step (1): The multiplier cards and the matrix cards are merged together and sorted on cols. (7–10) so that at the end of the operation, all the cards will be arranged in increasing order of row-numbers and each multiplier card is followed by corresponding matrix card having the same row and column numbers.

Step (2): Thus the cards will be divided into 'n' groups corresponding to 'n' row identifications. At the end of each group, a set of 'n' blank cards are filed; so that we will have a pack of $(3 \times n^2)$ cards arranged in the above specified way.

Step (3): The whole pack is now fed into the calculating punch, wherein the new multiplier values are developed and punched on the blank cards.

Step (4): After the calculation is over the three sets of cards are separated by sorting on the column showing the cycle number.

Step (5): Steps (1)—(4) are repeated using the new multiplier cards in place of the old multiplier cards, to obtain yet another new set of multiplier cards.

The whole procedure is stopped when $x_1^{(t+1)} = x_1^{(t)}$.

4. Programming on IBM Calculating Punch Type-626

Factor routing. (1) The value 'b_{ij}' is read from card columns 11-15 from matrix cards only (C.D.I. = 1). (2) The value 'm_{ij}' is read from card columns 16-20 from multiplier cards only (C.D.I. = 2). (3) C.D.I. = 2 is punched directly from the punch emitter on the new multiplier (blank) cards in card column 1. (4) Row numbers and column numbers are also punched in card columns 7-8 and 9-10 respectively. (5) The new multiplier values (m_{ij}) are punched in card columns 16-20. (6) The divisor x_1 is punched in card columns 21–27 on all the multiplier cards.

Read cycle operations. (1) From matrix card the information b_{ij} is read into Storage 6R. (2) From multiplier card the information m_{ij} is read into Storage 7L. (3) For all those matrix cards showing the elements $b_{1\,\overline{n-1}}, b_{2\,\overline{n-1}}, \ldots, b_{n\,\overline{n-1}}$, counters 4 and 5 are reset. (*Note* : On these cards X is punched in addition to the digit, in tens column of the field showing the column numbers by means of which a selector is picked up). (4) From all the multiplier cards showing the elements $m_{1n}, m_{2n}, \ldots, m_{nn}$, whatever information is there in the field allotted for the row number, is added to the first two positions of counter 5, which is transferred to the subsequent blank cards, on the field for column number. The cycle number from column 35 is also added in 3rd and 4th positions of counter 5, to which unity is added in the programme and the new cycle number is punched on the new multiplier cards.

Note : This is achieved by picking up a digit selector for the unit position of the number 'n'. It is to be noted that though the counter may add more than once, it retains only that information taken from the above multiplier cards for punching on the subsequent blank cards.

All these complicated ways of selection can be avoided, if we can have at least one more digit selector on the IBM-626 calculating punch.

Programme 1 *is picked up for multiplier* (m_{ij}) *cards* (C.D.I. = 2).

Programme 1 : (1) RISU Storage 6 so that Storage 6R is made ready to receive information (b_{ij}) from the matrix card. (2) Non-Punch.

Programmes 2 *and* 3 *are picked up for matrix* (b_{ij}) *cards* (C.D.I. = 1).

Programme 2 : (1) Read out Storage 6R to counters (1+2+3) and Storage 7L to multiplier unit 1. (2) Add in counters (1+2+3). (3) MPLY/DIVIDE Start 5. At this stage the sum of products ($b_{ij} \times m_{ij}$) is developed and is accumulated in counters (1+2+3).

Programme 3 : (1) RISU Storage 7, so that Storage 7L is ready to receive information (m_{ij}) from the multiplier cards. (2) Non-Punch.

Programmes 4 *to* 7 *are picked up for the matrix card showing the element* b_1n.

Programme 4 : (1) STRI.

Programmes 5 *and* 6 (coupled) : (1) Pick up co-selector 4. (2) Add 5 in counters (1+2+3) selectively in this programme through the above co-selector. Half adjustment is done at this stage. (3) Add 1 in units (not the first) position of the quotient counter (7+8). This is the value of the new multipliers (m_{i1}) for $i = 1, 2, \ldots, n$. (4) RISU Storage 3 so that the storage is made ready to receive (in the next programme) the information $x_1 = (b_{11}m_{11} + b_{12}m_{12} + \ldots + b_{1n}m_{1n})$, the constant divisor. (5) RISU Storage 4 so that the storage is also made ready to receive (in the next programme) the constant divisor, to be punched later on, on the blank cards. (6) RISU Storage 5 so that the storage is ready to receive (in the next programme) the quotient values, (which are the rest of the multiplier values) m_{ij} for $i = 1, 2, \dots, n$ and $j = 2, 3, \dots, n$. (7) Add 1 selectively to 3rd position of counter 5. Thus the previous cycle number which is already there in the counter is increased by unity.

Programme 7 : (1) Read out counters (1+2+3) to Storages 3 and 4. (2) Read out counters (7+8) to Storage 5 and reset. (3) Reset counters (1+2+3). (4) STRI.

Programmes 8 *to* 11 *are picked up for all those matrix cards showing the elements* b_{in} *for* $i = 2, 3, \dots, n$.

Programme 8 : (1) STRI.

Programme 9 : (1) Read out Storage 3 to counters (1+2+3). (2) Minus in counters (1+2+3). (3) MPLY/DIVIDE Start and Divide. (4) Plus in counters (7+8).

At this stage the quotient values are developed in counters (7+8).

Programme 10 : (1) Pick up co-selector 5. (2) Plus (Add 5) in counters (7+8). Half adjustment is done. (3) Plus (Add 1) in 3rd position of counter 5. The previous cycle number is increased by unity. (4) RISU Storage 5 so that the storage is made ready to receive (in the next programme) quotient values from counters (7+8).

Programme 11: (1) Read out counters (7+8) to Storage 5 and reset. (2) Reset counters (1+2+3). (3) STRI.

Programmes 12 *to* 14 *are picked up for blank cards.*

Programme 12 : (1) Pick up co-selector 8. (2) Plus (Add 1) in counter 4. (For each blank card unity will be added so that on the first blank card 1, as row number is punched, and on the second blank card 2, so on and on the n-th blank card n is punched as row number).

Programme 13 : (1) Read out Storages 4 and 5. (2) Read out counters 4 and 5. (3) Punch through Storages 1 and 2. (4) STRI.

Programme 14 : (1) RISU Storage 7.

5. Comments

The method is useful for matrices of any order for $n \leqslant 99$, of course depending on the number of digits of each element and counter capacity. Such computations can of course be done very quickly and efficiently on modern electronic computers, but the method developed here is much better than manual methods of computation. This procedure takes, remarkably less time, about ten minutes for each cycle for a matrix of size 15×15. The operational procedure is extremely simple, requires only one 626 control panel. The merging of cards in the specified way is done just with the help of a sorter, and all the information, including the cycle number is punched automatically by the calculating punch.

ACKNOWLEDGEMENT

My thanks are due to Dr. J. Roy, who has kindly suggested the problem, for his valuable guidance and constant encouragement. I also thank Sri A. Haldar, for his helpful suggestions in the preparation of this paper.

REFERENCES

ROY, J., CHAKRAVARTI, I. M. and LAHA, R. G. (1956): *Hand-Book for Practical Work in Statistics*, (mimeo), Research and Training School, Indian Statistical Institute, Calcutta.

IBM CORPORATION (1959): *Principles of Operations of Electronic Calculating Punch Type-626.* International Business Machine Corporation, New York.

Paper received : June, 1963.