JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Matrix Methods in Analysis Of Program And Measurements :

*A **M**etric based on the **E**igen Values Of **S**ource code **H**ierarchal **A**djacency Matrix* **(MESHA)**

[1]Daya Krishan Lobiyal [2]A.Meshach Ponraj, ,

[1]Professor,SCSS, Jawahar Lal Nehru University New Delhi- 110 067 ,
[2] Associate Professor DCS/SIT, Madurai Kamaraj University Maduari- 625 021
INDIA

Abstract : **:** A century ago matrix methods emerged as an idea to solve linear algebra problems as introduced by a nineteenth century mathematician James Sylvester along with his friend Arthur Cayley to develop algebraic aspects of matrices. Basically, we consider an array of numbers or symbols as marix which are arranged in fixed row and columns. Our approach seems to be a first attempt to introduce it to software metrics This study has been undertaken to introduce new metric which we call it 'MESHA', an acronym for a **M**eric based on the **E**igen Values of **S**ource code **H**ierarchal **A**djacency Matrix.

IndexTerms **– Software Metrics, Program Complexity, Maintenance of software.**

I. INTRODUCTION

The control structure that follows the execution of a program influences its complexity, it is logical to know that the more decisions the logic has to evaluate to know which way to go, it becomes more difficult to analyze or understand. With this intuition the Cyclomatic Number was introduced [1,2] This methodology proposed by McCabe measures the complexity of a program taking as a reference its control graph, the metric of this method is based on the cyclomatic number V (G) that is defined for a given graph, assuming that there is a corresponding graph to the flow of control of a program, with and e edges, *n* nodes and *c* connected components (usually c is 1). The cyclomatic complexity of this program will be given by the following formula :

command can be executed immediately after the first.

M = E -N +2 (1) Where:

M = Cyclomatic Complexity

E = Number of Connectors

N = Number of Nodes.

The cyclomatic number is the minimum number of paths necessary to construct any other path present in the graph through combinations, understanding as path a succession of nodes that can be traversed by following arcs (or connections between nodes) present in the graph.For example, in the directed graph given in the following figure where e = 12, n = 10 we have a cyclomatic complexity of 4, that is, *V (G) = 12 - 10 + 2 * 1.*

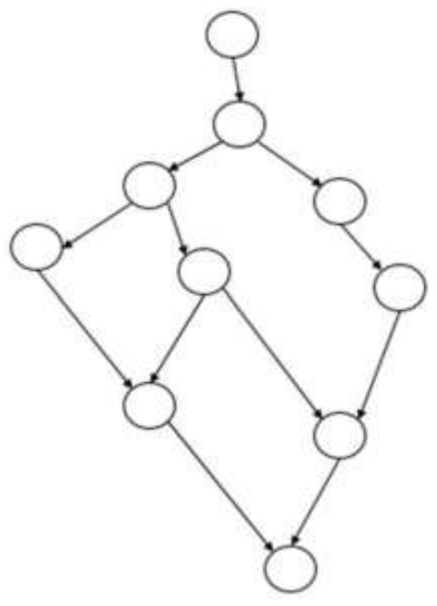


Figure1- A control flow graph with12 edges and 10 edges

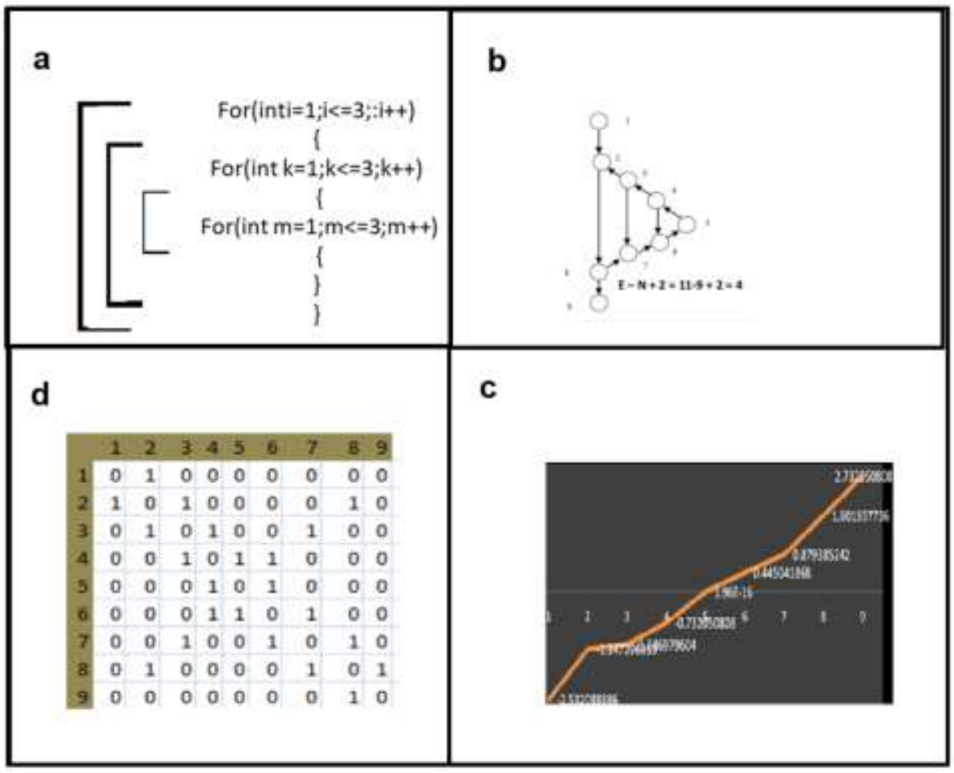
Cyclomatic complexity (or conditional complexity) is measured by the number of independent execution paths from a source code's flow chart. Graph nodes correspond to indivisible groups of commands. A connector links two nodes together if the second

From the measurement of cyclomatic complexity, McCabe (1976) seeks to capture a quantitative basis to perform the modularization of *software* that will be difficult to test or maintain. As we have said earlier, the cyclomatic complexity of a section of the source code is the count of the number of linearly independent paths through the source code [3 4]. For example, if the source code did not contain decision points such as IF statements or FOR loops, the complexity would be 1, since there is only one path through the code. If the code had a single IF statement containing a single condition, there would be two paths through the code, a path where the IF statement is evaluated as TRUE and a path where the IF statement is evaluated as FALSE. It is also used as a benchmark for comparison of different source codes.

It is observed that a program with more complexity is more likely to be error prone and hence it is difficult to maintain. A program with simple straight line execution without conditions or iterations is definitely easy to test . But such programs rarely exist.

We consider three types of loops. A simple IF-Else with one two, nested loop and a sequential loop.

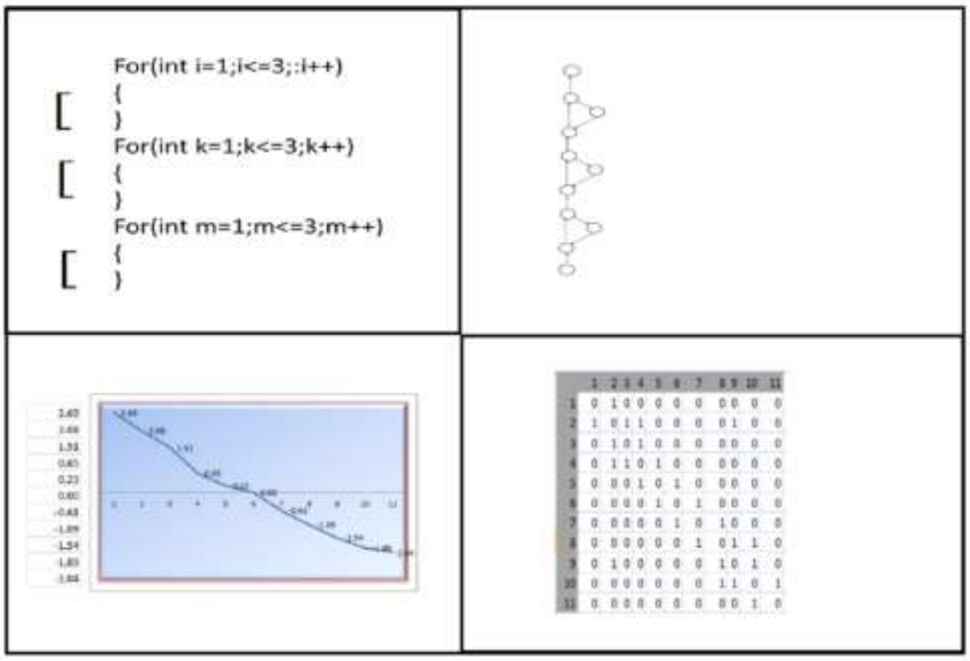
Our objective is to design a to differentiate among the three cases. The problem was solved in a ratio manner in such a way, that the lower bound represented number of decision statements plus one and the upper bound represented individual conditions plus on study addresses the problem by comparing two programs. One with a sequence of two loops and the other containing two loops in a nested fashion. The positioning of a control structure inside another control structure is called nesting e.g. nested-IF and nested-loop. Cyclomatic complexity fails to calculate complexity for nested structures. Many authors tried to solve this problem, and many quite successfully did to some extent. They argue that a nested-IF nested-loop is more complex than that of a sequential structure y a nested loop is considered to be more complex than a sequence of loops



```
For(inti=1;i<=3;:i++)
{
For(int k=1;k<=3;k++)
{
For(int m=1;m<=3;m++)
{
}
}
```

	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	1	0
3	0	1	0	1	0	0	1	0	0
4	0	0	1	0	1	1	0	0	0
5	0	0	0	1	0	1	0	0	0
6	0	0	0	1	1	0	1	0	0
7	0	0	1	0	0	1	0	1	0
8	0	1	0	0	0	0	1	0	1
9	0	0	0	0	0	0	0	1	0

Figure 2 - *Program Text analysis of a nested loop-Clock wise from top right- figure a) a represents ta control flow structure of a nested loop . b) Represents its CFG. The Cyclomatic complexity is 4 c) an incident matrix of GCF d) computed eigenvalues*



```
For(int i=1;i<=3;:i++)
{
}
For(int k=1;k<=3;k++)
{
}
For(int m=1;m<=3;m++)
{
}
```

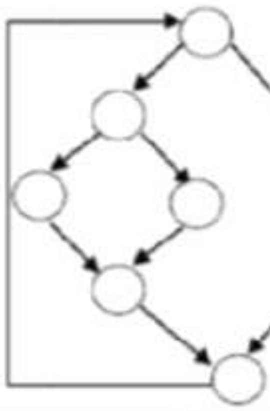
	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	0	0
2	1	0	1	1	0	0	0	0	1	0	0
3	0	1	0	1	0	0	0	0	0	0	0
4	0	1	1	0	1	0	0	0	0	0	0
5	0	0	0	1	0	1	0	0	0	0	0
6	0	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0
8	0	0	0	0	0	0	1	0	1	1	0
9	0	1	0	0	0	0	0	1	0	1	0
10	0	0	0	0	0	0	0	1	1	0	1
11	0	0	0	0	0	0	0	0	0	1	0

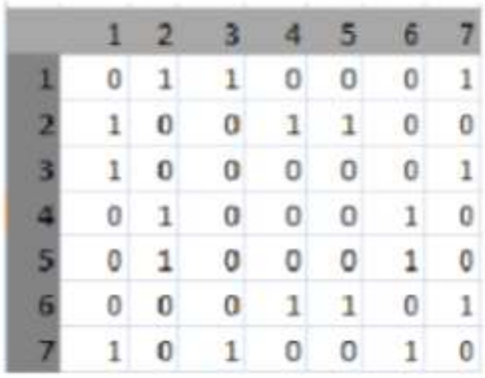
The CFG (Control flow graph) of the code is also given in the above figure. The incident matrix is represented as follow- we have made it real symmetric in order to avoid eigenvalues in the complex domain/ negative values.

Figure -3:a program text analysis of a sequential loop with GFC,Adjacency matrix and its eigenvalues

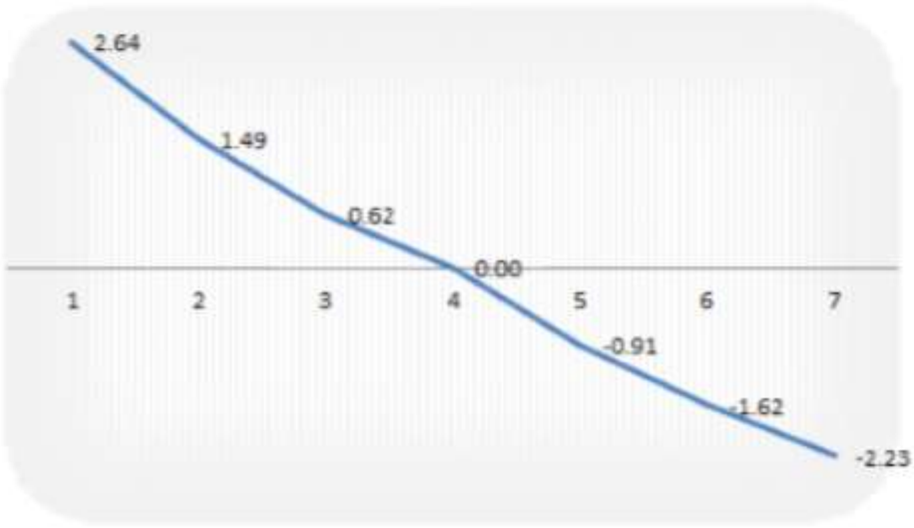
Here we have 11 edges and 9 nodes. So the cyclomatic complexity is N+2=11-9+2 = 4 In the next section let us analyse a partially nested if with a sequential if condition

In the CFG the node 1 and two are connected. So in the matrix the (1,2) and (2, 1) has entry as '1'.





	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	0	1	1	0	0
3	1	0	0	0	0	0	1
4	0	1	0	0	0	1	0
5	0	1	0	0	0	1	0
6	0	0	0	1	1	0	1
7	1	0	1	0	0	1	0



We have computed the eigen values using MATLAB and plotted them as follows. As it is a 9X9 matrix, we have 9 eigen values. The maximum of the eigen value is 2.73. Thus the complexity of the above source code is 2.73 Mesha(CN) because it is based on the CFG with nested loop. The eigen value 0.000000000000000001 9 is represented as 1.96E-16.

Figure 4 Program text analysis of a partially nested loop

Without loss of generality this may be treated as zero.Thus Mesha is very sensitive to nested loop than a sequential loop. In other words, a nested loop is more complicated than a sequential loop.

Norm -1: It is defines as follows**:**

$$\|A\|_1 = \max_j \left(\sum_i |a_{ij}|\right)$$

This is nothing but the maximum of the column/row sum of absolute values of each elememt. We can explain pictorially as follows:

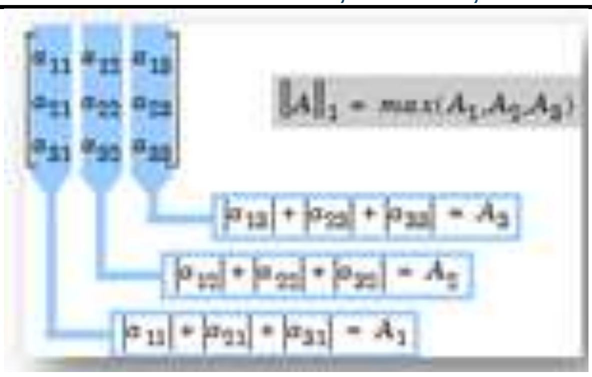

a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}
$\|A\|_1 = max(A_1,A_2,A_3)$
$|a_{13}| + |a_{23}| + |a_{33}| = A_3$
$|a_{12}| + |a_{22}| + |a_{32}| = A_2$
$|a_{11}| + |a_{21}| + |a_{31}| = A_1$

Figure 5- Computation of Norm-1

Columns and rows can be used interchangeably because we have made our matrix as symmetric.
Norm -2: It is defined as square root the sum of squares of each elements.

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

Subordinate to the vector 2-norm is the matrix 2-norm

$$\|A\|_2 = \sqrt{\text{largest eigenvalue of } A^*A} \ .$$

In our case all a_{ij} are zeros and ones. So A*A = A. So it is nothing but the square root of sum of all elemets. In others words square root of edges. Thus the cyclomatic complexity can be rewritten as:
M = $(\text{Norm-2})^2$ –(size of adjacency matrix) +2 because E = $(\text{Norm-2})^2$ The Frobenius norm requires that we cycle through all matrix entries, add their squares, and then take the square root. This involves an outer loop to traverse the rows and an inner loop that forms the sum of the squares of the entries of a row. It is defined for a MXN matrix. In our case M=N. for all matrix element $a_{i,j}$

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

So The cyclomatic complexity is, $M = [\|A\|]^2 - \text{Size of Matrix} + 2$

Table 1 Comparison of Various metrices

Metrics	**Partially Nested if**	**Sequentially if**	**Sequentially Loop**	**Nested Loop**
Mesha	2.6426	2.6813	2.6941	2.7321
Norm-1	3	1	4	3
Norm-2	2.6426	2.6813	2.6941	2.7321
Norm-Fro	4.2426	4.2426	5.099	4.6904
Cyc.Comp	4	4	4	4
Edges	9	9	13	11
Nodes	7	7	11	9

Summary and conclusion:

All existing software metrics are defined in the domain of integers. They are not sensitive to nesting and looping. We have defined a set of metrics which are in the domain of real numbers with few significant digits as marked by dark space with white dots. The metrics which we introduced are not competing with the existing metrics systems but they are complementing to the existing system of metrics. The metrics norm lies on the boundaries. From table 1 it is observed that the 'MESHA' is very sensitive to nesting and looping

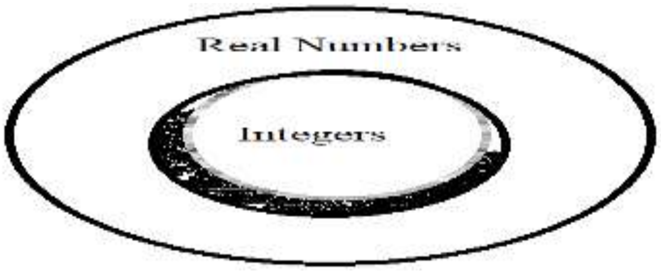


Figure 5-Software complexity metric extended to domain of real numbers

Bibliography

1. [Laplante 11] Phillip A Laplante," Encyclopaedia of Software Engineering", CRC Press FL ©2011.
2. [Mukt 20] Mukt Shabd. "Maintability Index Computation Based on cyclomatic Complexity and Halstead Measures" IX(X): 699–709. http://shabdbooks.com/gallery/75-oct2020.pdf
3. [Nesc 20] Nesc, Michael D Squire et al. "Cyclomatic Complexity and Basis Path Testing Study." (December 2020). https://ntrs.nasa.gov/api/citations/20205011566/downloads/20205011566.pdf
4. [Shepperd 88] Shepperd, Martin, 'A Critique of Cyclomatic Complexity as a Software Metric' Software Engineering Journal 1988/04/01 DO - 10.1049/sej.1988.0003 https://www.researchgate.net/publication/3407068_A_Critique_of_Cyclomatic_Complexity_as_a_Software_Metric/citation/download