Achieving Capability in Data Compression Using LZW++ Technique

Yusof. Mohd Kamir
mohdkamir@udm.edu.my
Universiti Darul Iman Malaysia
Kampus Kusza, Gong Badak
21300 Kuala Terengganu
Terengganu, Malaysia
Tel: +609 6653352
Fax: +609 6673412

Mat Deris. Mohd Sufian
sufian@udm.edu.my
Universiti Darul Iman Malaysia
Kampus Kusza, Gong Badak
21300 Kuala Terengganu
Terengganu, Malaysia
Tel: +609 6653412
Fax: +609 6673412

Abidin. Ahmad Faisal Amri
faisalamri@udm.edu.my
Universiti Darul Iman Malaysia
Kampus Kusza, Gong Badak
21300 Kuala Terengganu
Terengganu, Malaysia
Tel: +609 6653414
Fax: +609 6673412

Madi. Elissa Nadia
elissa@udm.edu.my
Universiti Darul Iman Malaysia
Kampus Kusza, Gong Badak
21300 Kuala Terengganu
Terengganu, Malaysia
Tel: +609 6653411
Fax: +609 6673412

Abstract
The development of efficient compression software to compress text and image is a challenging task. This paper presents the enhanced LZW technique in data compression. The basic framework of enhanced LZW technique is based on existing LZW technique. Modification of existing LZW structure was done in order to produce the enhanced LZW technique. Enhanced LZW technique read three characters in time during data compression while existing LZW technique read character one by one. This project employs enhanced LZW technique on data compression with text and image. Experimental test have been done performed on text and image. Comparison was made between enhanced LZW technique and existing LZW technique in terms of time performance and size of file after compression. The experiment results show that enhance LZW technique is more efficient in text compression compare to existing LZW technique.

Key words:
Data Compression, LZW, LZW++, LZ-77, RLE, Text, Image.

1. INTRODUCTION

Data compression is often referred to as coding, where coding is very general term encompassing any special representation of data which satisfies a given need. It is useful because it helps to reduce consumption of expensive resources such as hard disk space or transmission bandwidth i.e. a data file that suppose to takes up 50 kilobytes (KB) could be downsized to 25 kilobytes (KB), by using data compression software. A simple characterization of data compression is that it involves transforming a string of characters in some representation (such as ASCII) into a new string (of bits, for example) which contains the same information but whose length is as small as possible. Data compression has important application in the areas of data transmission and data storage. Compressed data required smaller storage size and reduce the amount of data that need to be transmitted. Hence, it increases the capacity of the communication channel.

Several techniques have been used for data compression. Each technique has their advantages and disadvantages in data compression. In this paper, 4 types of data compression algorithm namely Run Length Encoding (RLE), Huffman Encoding, LZ-77 and Lempel Ziv Welch (LZW) were analyzed. One of the algorithms is chosen in this paper to study for potential improvement and enhancement.

2. PREVIOUS WORKS

One of the techniques for data compression is "run length encoding", which is sometimes knows as "run length limiting" (RLL) [5, 6]. Run length encoding is very useful for solid black picture bits. This technique can be used to compress text especially for text file and to find the repeating string of characters. This compression software will scan through the file to find the repeating string of characters, and store them using escape character (ASCII 27) followed by the character and a binary count of he number of items it is repeated. This compression software must be smart enough not to compress strings of two or three repeated characters but more than that. Instead, if the compression software is not smart, this technique will produce the bigger size than original size. First problem with this technique is the output file is bigger if the decompressed input file includes lot of escape characters. Second problem is that a single byte cannot specify run length greater than 256.

Another technique for data compression is know as "Huffman coding" [4]. In this encoding technique, characters in a data file are converted to a binary code, where the most common characters in the file have the shortest binary codes, and the least common have the longest. In Huffman coding also, the assignment of codewords to source messages is based on the probabilities which the source messages appear in the message ensemble. Huffman are used to compress and decompress different type of files such as doc, bmp, jpg, tiff, tif and gif. Bmp files contain images, in which each dot in the image represented by a byte. However, the problem with this technique is text compression.

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

Another technique for data compression is LZ-77 encoding [7]. This technique is a simple, clever and effective approach to compress text. This technique exploits the fact that words and phrases within a text stream are likely to be repeated. When they repeat, they can be coded as a pointer to an earlier occurrence, with the pointer accompanied by the number of characters to be matched. This technique is useful for compressing text because it able to reduce the file size and increase the compression ratio after compression. However, it is not efficient for image file format such bmp, gif, tif and tiff. Beside that, this technique will take several minutes to compress a data. Sometimes, the long processing time will cause the missing of some characters.

The most popular technique for data compression is Lempel Ziv Welch (LZW) [8]. LZW is a general compression algorithm capable of working on almost any type of data. It is generally fast in both compressing and decompressing data and does not require the use of floating-point operations. LZW technique also has been applied for text file. This technique is very efficient to compress image file such tiff and gif. However, this technique not efficient for compress text file because it require many bits and data dictionary.

Based on several techniques for data compression, LZW technique produce better result compared to RLE and Huffman Coding. LZW have a great potential to be improved in order to produce a better result than existing LZW technique in term terms of data size and time performance.

3. RESEARCH METHODOLOGY

This paper shows comparison process between existing LZW and enhanced LZW techniques.

3.1 LZW Algorithm

Fig 1 shows the compress algorithm and fig 2 show the decompress algorithm for LZW technique.

```
L1    Start
L2    S = get input character
L3    WHILE there are still input character not equal to null
L4    IF S + C is in the string table THEN
                S = S + C
L5    ELSE
                Output the code for S
                Add S + C to the string table
                S = C
      END IF
L6    END of WHILE
      Display Output
```

Fig. 1 Compress Algorithm

```
L1    Read Old_Code
L2    Output Old_Code
      WHILE there are still input characters D0
                Read NEW_CODE
                String = get transaltion of NEW_CODE
                Output String
                Character = first character in STRING
                Add OLD_CODE + CHARACTER to the
                translation table
L3    OLD_CODE = NEW_CODE
      END of WHILE
```

Fig. 2 Decompress Algorithm

Example 1 show compression process and example 2 show the decompression process on data compression using existing LZW technique.

Example 1

String: before compression **PPPQPPQQQ**

Table 1: Compress Dictionary 1

Code	0	1
Key	P	Q

0 PPQPPQQQ – Table 1 shows character **P** represent by **0**. That way, character **P** replace by **0**.

Table 2: Compress Dictionary 2

Code	0	1	2
Key	P	Q	PP

0 2 QPPQQQ – Table 2 shows character **PP** represent by **2**. That way, character **PP** replace by **2**.

Table 3: Compress Dictionary 3

Code	0	1	2	3
Key	P	Q	PP	PPQ

0 2 1 3 QQ – Table 3 shows character **PPQ** represent by **3**. That way, character **PPQ** replace by **3**.

Table 4: Compress Dictionary 4

Code	0	1	2	3
Key	P	Q	PP	PPQ

0 2 1 3 1 Q – Table 4 shows character **Q** represent by **1**. That way, character **Q** replace by **1**.

Table 5: Compress Dictionary 5

Code	0	1	2	3
Key	P	Q	PP	PPQ

0 2 1 3 1 1 – Table 5 shows character **Q** represent by **1**. That way, character **Q** replace by **1**.

Table 6: Compress Dictionary 6

Code	0	1	2	3
Key	P	Q	PP	PPQ

Finally, string **PPPQPPQQQ** compressed to **021311**. Original size for string **PPPQPPQQQ** is 9 x 8 byte (size ASCII) = 72 bytes. After compressed, string **PPPQPPQQQ** change to **021311** and size after compress is 6 x 4 byte = 24 bytes.

Example 2
String: before decompress **021311**

Table 7: Decompress Dictionary 1

Code	0	1	2	3
Key	P	Q	PP	PPQ

Step 1:
P 21311 – Table 7 shows character **0** represent by **P**. That way, character **0** replace by **P**.

Step 2:
PPP 1311 – Step 2 shows character **2** represent by **PP**. That way, character **2** replace by **PP**.

Step 3:
PPP Q 311 – Step 3 shows character **1** represent by **Q**. That way, character **1** replace by **Q**.

Step 4:
PPP Q PPQ 11 – Step 4 shows character **3** represent by **PPQ**. That way, character **3** replace by **PPQ**.

Step 5:
PPP Q PPQ Q1 – Step 5 shows character **1** represent by **Q**. That way, character **1** replace by **Q**.

Step 6:
PPP Q PPQ QQ – Step 6 shows character **1** represent by **Q**. That way, character **1** replace by **Q**.

After decompress, string **021311** changes to original string. The original character is **PPPQPPQQQ**.

3.2 Enhanced LZW Algorithm

Enhancement LZW algorithm is needed to produce better result compare to existing LZW algorithm. This enhanced LZW algorithm read the string and divide the string to smaller string which consist of three characters until end of the string. Fig 3 shows the algorithm for compress and fig 4 shows the algorithm for decompress data.

```
L1   Start
L2   String = Get Input Character
L3   Character = Get Input Character
     Assign Count = 0
     Assign i = 0; j = 0
L4   WHILE there are still input character not equal to
     Null
L5   Variable A = Character[j]
     Count++;  B = B*A; j++
L6   IF (count = = 3)
L7   IF (B !=Code)
                Code = B
                Key = i
                i++
     ELSE
                B = Key
     END IF
L8   C = C * Key
     j = j
     count = 0
L9   Repeat Line L4
```

```
        ELSE
L10     Repeat Line L4
        END IF
        END of WHILE
        Display Output
        END`
```

Fig. 3 Compress Algorithm

Algorithm for compression can be represented in formula as below:

$X = \{x_1, x_2, x_3, x_4, x_5, \ldots\ldots\ldots, x_n\}$

Where x represents the whole string, x1 represent the first character in the string, x2 represent the second character, and so on. Even though in reality the length of the string is finite, mathematically assume that it has infinite length.

$$H = \log_2 m \text{ bits/character} \quad (1)$$

$$H = \sum_{i=1}^{m} pi \log_2 p_i \text{ bits/character} \quad (2)$$

General model: Let B_n represent the first n characters. The entropy rate in the general case is given by

$$H = \lim_{n\to\infty} \frac{1}{n} \sum p(Bn) log2\ p(Bn) bits/character \quad (3)$$

Where the sum is over all possible values of B_n. It is virtually impossible to calculate the entropy rate according to the above equation. Using a prediction method, Shannon has been able to estimate that the entropy rate of the n letter.

```
L1      Start
L2      Read String
L3      Assign Character = String
        i = 0
L4      WHILE there are still input character not equal to null
L5      Read Dictionary
L6      Character[i] = Code
L7      A = A * Character[i]
        i++
        Repeat Line L3
L8      END of WHILE
L9      Display Output
L10     END
```

Fig. 4 Decompress Algorithm

Example 1 and example 2 show the compression and decompress process using enhanced LZW algorithm and. This technique will read the string and read the character one by one. After that, the character will represent by key.

Example 1

String: before compression **PPPQPPQQQ**

Table 8: Compress Dictionary 1

Code	0
Key	PPP

0 QPPQQQ – Table 8 shows first three characters represent by **0**. Characters **PPP** already replaced by **0**.

Table 9: Compress Dictionary 2

Code	0	1
Key	PPP	QPP

0 1 QQQ– Table 9 shows the second set of three characters represent by **1**. **QPP** replaced by **1**.

Table 10: Compress Dictionary 3

Code	0	1	2
Key	PPP	QPP	QQQ

0 1 2 – Table 10 shows the last three characters represent by **2**. That, character **QQQ** replaced by **2**.

Finally, string **PPPQPPQQQ** is compressed to **012**. Original size for string **PPPQPPQQQ** is 9 x 8 byte (size ASCII) = 72 bytes. After compressed, string **PPPQPPQQQ** change to **012** and size after compress is 3 x 4 byte = 12 bytes. The string size has decrease be cleared from 72 bytes.

Example 2

String: before decompression **012**

Table 11: Decompress Dictionary 1

Code	0
Key	PPP

PPP 12 – Table 11 shows first characters represent by **PPP**. That way, character **0** replace by **PPP**.

Table 12: Decompress Dictionary 2

Code	0	1
Key	PPP	QPP

PPPQPP 2– Table 12 shows the second characters represent **QPP**. That way, character **1** replace by **QPP**.

Table 13: Decompress Dictionary 3

Code	0	1	2
Key	PPP	QPP	QQQ

PPPQPPQQQ – Table 13 shows the last character represent by **QQQ**. That, character **2** replace by **QQQ**.

Finally, string **012** decompress to **PPPQPPQQQ**. Original size before decompress is **012** is 3 x 4 byte (size ASCII) = 12 bytes. After decompress, string **012** change to **PPPQPPQQQ** and size after decompress is 9 x 8 byte = 72 bytes.

4. EXPERIMENTAL RESULT

The data set are divided into two segment are text and image. Different data types will used for experiment to get the result which presented by graph.

4.1 Measurement

In this experiment, measurement is important to be considered. Basically, measurement divided into 2 parts;

File size - is the size of a computer file. File size can be kilobytes (Kb), megabits (Mb) and so on. The same file size of data which use in LZW technique should be used for carry out the experiment on enhancement LZW technique.

Time for Compress - Another measurement is time performance for compress the data. Time used for compress a data using enhanced LZW technique will be compared with the time used for compress same data using existing LZW technique.

4.2 Input Parameter

Before carry out the experiment, the most important thing is the consideration about parameters. The parameter must fulfill the criteria to produce the result after compression process. Table 13 below shows the explanation of input parameters.

Table 13: Explanation of Input Parameter

Data Type	**Data Format**	**Data Size**	**Explanation**
Text	.doc, .pdf, .txt	Range between 1kb to 500kb	the same data format will used for experiment. Evaluation divided to two part, after compressed and time performance.
Image	.gif, .bmp	Range between 1kb to 500kb	image will used for experiment and the evaluation based on after compressed size and time performance.

4.3 Experiments

The experiments used different data types as input, the results generated by the enhanced LZW are compared with the result generated by existing LZW. Total of 8 experiments been carry out, results from each experiments are shown in this section.

Experiment 1

Experiment 1 used text as data type (document type). This experiment will show the result file size after compression using existing LZW technique and enhanced LZW technique.

Fig 5 shows the comparison between existing LZW technique with enhanced LZW technique in term of file size.

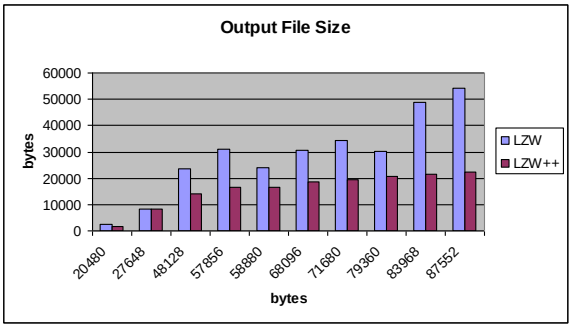

Fig. 5 Comparison output file size between LZW and LZW++

Experiment 2

Experiment 2 used text as data type (document type). This experiment will compare the results between LZW

techniques with enhanced LZW technique in term of time performance for compression.

Fig 6 shows the comparison between existing LZW technique with enhanced LZW technique in term of time performance.

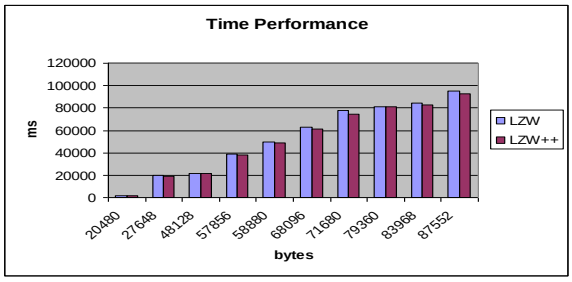

Fig. 6 Comparison of time for compress between LZW and LZW++

Experiment 3

Experiment 3 used text as data type (txt type). This experiment will show the result file size after compression using existing LZW technique and enhanced LZW technique.

Fig 7 shows the comparison between existing LZW technique with enhanced LZW technique in term of file size.

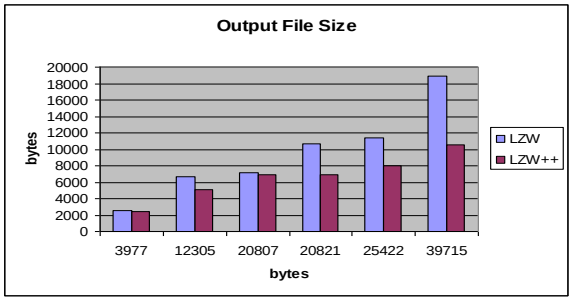

Fig. 7 Comparison of file size between LZW and LZW++

Experiment 4

Experiment 4 used text as data type (txt file). This experiment will compare the results between existing LZW technique with enhanced LZW technique in term of time performance for compression.

Fig 8 shows the comparison between existing LZW technique with enhanced LZW technique in term of time performance.

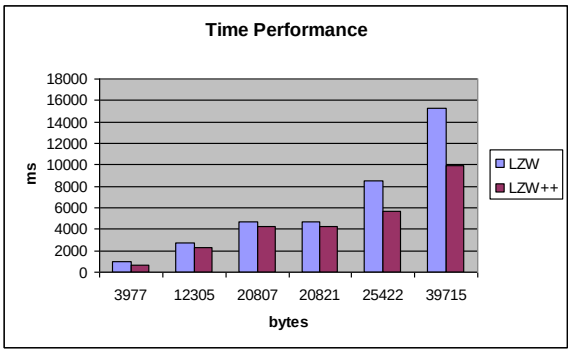

Fig. 8 Comparison of time for compress between LZW and LZW++

Experiment 5

Experiment 6 used image as data type (bmp file). This experiment shows the result file size after compression using existing LZW technique and enhanced LZW technique.

Fig 9 shows the comparison between existing LZW technique with enhanced LZW technique in term of file size.

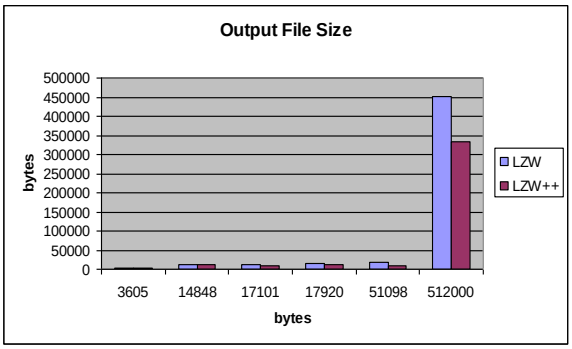

Fig. 9 Comparison of file size between LZW and LZW++

Experiment 6

Experiment 6 used image as data type (bmp file). This experiment will compare the results between existing LZW technique with enhanced LZW technique in term of time performance for compression.

Fig 10 shows the comparison between existing LZW technique with enhanced LZW technique in term of time performance.

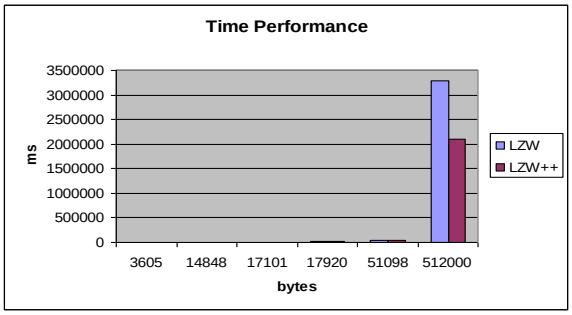

Fig. 10 Comparison of time for compress between LZW and LZW++

Experiment 7

Experiment 7 used image as data type (gif type). This experiment will show the result file size after compression using existing LZW technique and enhanced LZW technique.

Fig 11 shows the comparison between existing LZW technique with enhanced LZW technique in term of file size.

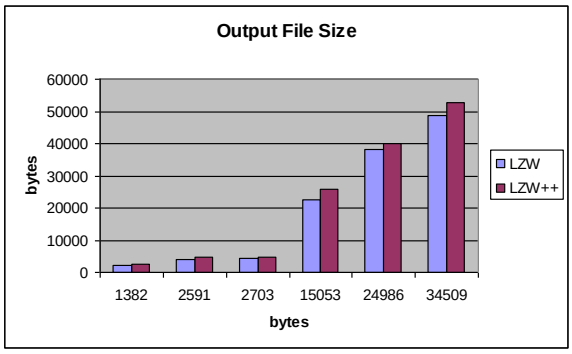

Fig. 11 Comparison of file size between LZW and LZW++

Experiment 8

Experiment 8 used image as data type (gif file). This experiment will compare the results between existing LZW technique with enhanced LZW technique in term of time performance for compression.

Fig 12 shows the comparison between existing LZW technique with enhanced LZW technique in term of time performance.

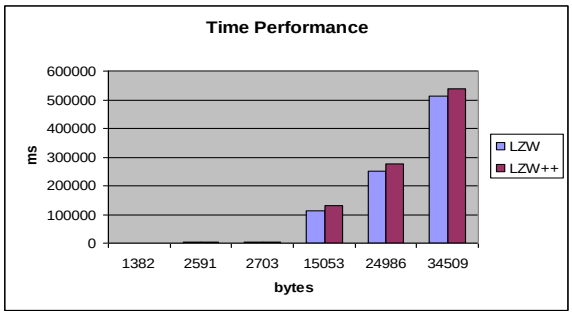

Fig. 12 Comparison of time for compress between LZW and LZW++

Based on the result produce by experiment 1 and 3, enhanced LZW technique is better than existing LZW technique in term of compressed file size. Both of these experiment used the same data type (i.e : text) and same data format (i.e : .doc and .txt). Based on result represented by graph, the blue bar represent results from existing LZW and red bar represent result from enhanced LZW technique. Experiment 2 and 4 produce the result for data compression in term of time performance. The results also, show that the enhanced LZW technique is better than LZW technique. In summary, enhanced LZW technique is very useful for compress text and be able to produce better result compared to existing LZW technique in terms of size and time performance.

Experiment 5 and 7 used the image as data type. Experiment 5 used image in bmp format while experiment 7 used image in gif format. Experiment 6 and 8 used the same data type but comparisons are done in term of time performance. Based on result produce by experiment 5 and 7, enhanced LZW technique is better than existing LZW technique in term size for image format bmp. As for image in gif format, enhanced LZW technique produce bigger file size after compress compared to existing LZW technique. In summary, enhanced LZW technique better for image format bmp but not for image format gif.

5. Conclusion

This work demonstrates a methodology for compress and decompresses data using enhanced LZW technique. The modifications needed to make existing LZW algorithm to produce new algorithm in order to produce better result than existing LZW technique. Therefore, several experiments are carried out in order to show performance enhanced LZW technique compared with LZW technique. Several tasks that have been done are evaluating performance of enhanced LZW technique with using different data type, choosing the best data type for enhanced LZW technique and comparing the performance and effectiveness of enhanced LZW technique with existing LZW technique in term of data size and time for compress.

Finally, the enhanced LZW technique is useful for text such as file format doc, pdf and txt. Based on the experimental result have been done, it achieve better result compared to existing LZW technique. Overall, the result also show that enhanced LZW technique have the potential to be used in compressing text.

References

[1] Ian, H.H. Witten, Moffat, a. and Bell, T.C. (1999). Managing Gigabytes: Compressing and Indexing Documents and Images.

[2] Vilter, J.S. (1987). Design and Analysis of Dynamic Huffman Codes. J. ACM 34, 4 : 825-845

[3] Bentley, J.L., Sleator, D.D., Tarjan, R.E., and wei, V.K. (1986). A Locally Adaptive Data Compression Scheme. Commun. ACM 29, 4 (Apr.), 320-330.

[4] Snowbirh, U. (2000). Data Compression Conference (DDC'00.)

[5] Mohmmad Al-laham and Ibrahim M.M. EL Emary. (2007). Comparative Study Between Various Algorithm of Data Compression Technique. 284-290.

[6] Gilbert, H. (1996). Data Image Compression Tools and Technique. WILEY : 68-343.

[7] Ngoc, V. and Alistair, M. (2006). Improved wordaligned binary compression for text indexing. IEEE Trans. Knowledge & Data engineering, 18: 857-861.

[8] Gawthrop, J. and W. Liuping, (2005). Data Compression for estimation of the stable and unstable linear systems. Automica, 41: 1313-1321.

[9] Kesheng, W., J. Otoo and S. Arie, (2006). Optimizing bitmap indices with efficient compression. ACM Trans. Database System.

[10] Julia, C.B. and Anita, C.M (2002). Programming in Visual Basic 6.0. Update edition. Avenue of the Americas, N. Y.: McGraw-Hill. 2-3.

[11] Blelloch, E. (2002). Introduction to Data Compression. Computer Science Department, Garnegie Mellon Universitiy.

[12] Gallager, R.G. (1978). Variations on a Theme by Huffman. IEEE Trans. Inform. Theory: 668-674.

[13] Gawthrop, J. and Luiping W. (2005). Data Compression for estimation of the physical parameters of stable and unstable linear system. Automatice, 41: 1313-1321.

Mohd Kamir Yusof obtained her Master of Computer Science from Faculty of Computer Science and Information System, Universiti Teknologi Malaysia in 2008. Currently, he is a Lecturer at Department of Computer Science, Faculty of Infomatics, Universiti Darul Iman Malaysia (UDM), Terengganu, Malaysia.

Mohd Sufian Mat Deris obtained her Master of Education (educational technology) from Faculty of Education, Universiti Teknologi Malaysia in 2006. Currently, he is a Lecturer at Department of Multimedia, Faculty of Infomatics, Universiti Darul Iman Malaysia (UDM), Terengganu, Malaysia.

Ahmad Faisal Amri Abidin obtained her Master of Computer Science from Faculty of Computer Science and Information Technology, Universiti Putra Malaysia in 2008. Currently, he is a Lecturer at Department of Computer Science, Faculty of Infomatics, Universiti Darul Iman Malaysia (UDM), Terengganu, Malaysia.

Elissa Nadia Madi obtained her Master of Mathematics from Faculty of Science and Technology, Universiti Malaysia Terengganu in 2009. Currently, he is a Lecturer at Department of Information Technology, Faculty of Infomatics, Universiti Darul Iman Malaysia (UDM), Terengganu, Malaysia.