INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND DEVELOPMENT

IJARnD

(Volume 4, Issue 8)

Available online at: www.ijarnd.com

Utilization of MATLAB as a technological tool for teaching numeric problems

Marlar Win Khin

University of Computer Studies, Banmaw, Myanmar

ABSTRACT

This paper illustrates how MATLAB can be used in numeric problems to enhance the teaching of Fixed-Point Iteration, Newton's Method, Secant Method, Gauss-Seidel Iteration and Least Squared Method. The education system is undergoing rapid changes. Various new methods are introduced and used. Further, it makes more effective and learning is highly significant. Today the use of MATLAB in teaching numerical analysis is widespread. MATLAB has grown into a comprehensive programming environment suitable for solving a wide range of numerical problems. The students can extend knowledge gained from the MATLAB course on the other courses. This paper explains the general process of numerical problems, studies the function and characteristic of MATLAB and its application in numerical problems.

Keywords— *MATLAB, Numeric problems, Education system, Numerical analysis*

1. INTRODUCTION

Today's software offers more for numerical analysis than just programming. The software MATLAB can be used to do things the traditional way; writing loops; branching using logical decisions and invoking subroutines. Now a larger programming environment is available; graphics and built-in subroutine libraries. These features are influencing the way numerical analysis is taught. MATLAB is based on lists of many algorithms can be streamlined by taking advantage of this structure.

MATLAB is a high-level language with interactive environment developed by MathWorks. It was designed to aid numerical computation, visualization and application development. It also allows matrix manipulations, plotting of functions and data, and interfacing with programs written in other languages like C, C++, Java, and FORTRAN.

A course in Numerical Methods studies the way we use computers to solve mathematically-based problems, a particularly important skill for students. Traditionally this means covering the theory and practice of numerically calculating typical mathematical [4]and numerical problems such as Fixed-Point Iteration, Newton's Method, Secant Method, Gauss-Seidel Iteration and Least Squared Method.

Our aim in developing the course numerical analysis was to give students some of the basic tools for solving numerical problems, while still giving them some experience in writing more complicated programs and a few of the numerical issued involved in computational methods. We try to teach them the in-built MATLAB commands, their uses, and limitations, as well as an introduction to programming their own simple routines. The advantage of giving students an exposure to MATLAB solving of problems, and the underlying numerical methods, is that these students can then use these skills for all their later courses, hence deepening their understanding of other topics. As an underlying principle, we wanted the course to be a numerical problem-focused since having a realistic application both motivates [5] students and gives them a deeper learning experience [6].

2. EXPERIMENTAL RESULTS OF NUMERIC PROBLEMS UTILIZING MATLAB

Some numerical examples that will be performed using MATLAB functions are illustrated. For the experiment in numerical analysis, some varied functions such as Fixed-Point Iteration, Newton's Method, Secant Method, Gauss-Seidel Iteration and Least Squared Method are demonstrated.

2.1 Solutions of Equations by Iteration

2.1.1 Fixed-Point Iteration for Solving Equations

By some *algebraic steps*, we transform $f(x) = 0$ into the form $x = g(x)$

Then we choose an x_0 and compute

$$x_1 = g(x_0), x_2 = g(x_1), ,$$

And in general

$$x_{n+1} = g(x_n)\ (n = 0, 1, \dots).$$

This solution is called a fixed point of g.

Example: An Iteration Process. Find a solution to

$$f(x) = x^3 + x - 1 = 0$$

By iteration.

Solution: We change the given equation to g(x),

$$x = g_1(x) = \frac{1}{1 + x^2}$$

$$x_{n+1} = \frac{1}{1 + {x_n}^2}$$

We choose x_0=1, we obtain

$$x_1 = 0.500, x_2 = 0.800, x_3 = 0.610, x_4 = 0.729, x_5 = 0.653, x_6 = 0.701, \dots$$

The solution exact to 6D is

$$s = 0.682328.$$

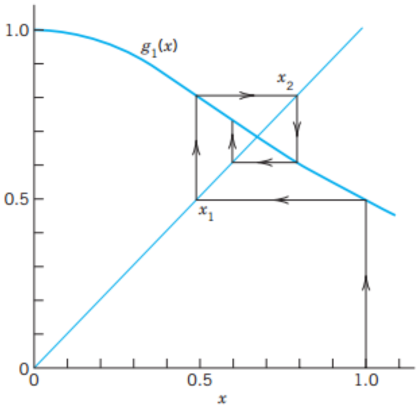


Fig. 1: Iteration of the above example

```
Editor - C:\Users\CUBMW\Marlarwinkhin\matlabprogram\fixpt1.m
fixpt1.m

function x = fixpt1 (g,x0,tol,n)

for i = 1:n % Do n times
    xi= g(x0); % Newton 's formula
    if abs(xi-x0)<= tol*abs(xi)
        break
    end
    x0=xi;
    fprintf('x(%d)=%9.6f\n',i,xi)

end
    fprintf('Ans=%9.6f\n',xi);
  if i==n
    fprintf('Solution was not obtained in %i iterations.\n',n)
    x=('no answer');
  end
```

Fig. 2: MATLAB Function of Fixed-Point Iteration

```
>> g=@(x)1/(1+x^2);
>> x0=1;tol=0.000001; n=50;
>> fixpt1(g, x0, tol, n)
x(1)= 0.500000
x(2)= 0.800000
x(3)= 0.609756
x(4)= 0.728968
x(5)= 0.653000
x(6)= 0.701061
x(7)= 0.670472
x(8)= 0.689878
x(9)= 0.677538
x(10)= 0.685374
x(11)= 0.680394
x(12)= 0.683557
x(13)= 0.681547
x(14)= 0.682824
x(15)= 0.682013
x(16)= 0.682528
x(17)= 0.682201
x(18)= 0.682409
x(19)= 0.682276
x(20)= 0.682360
x(21)= 0.682307
x(22)= 0.682341
x(23)= 0.682319
x(24)= 0.682333
x(25)= 0.682324
x(26)= 0.682330
x(27)= 0.682326
x(28)= 0.682329
x(29)= 0.682327
x(30)= 0.682328
Ans= 0.682328
```

If we change iteration number of n=10, we get

```
>> n=10;
>> fixpt1(g, x0, tol, n)
x(1)= 0.500000
x(2)= 0.800000
x(3)= 0.609756
x(4)= 0.728968
x(5)= 0.653000
x(6)= 0.701061
x(7)= 0.670472
x(8)= 0.689878
x(9)= 0.677538
x(10)= 0.685374
Ans= 0.685374
Solution was not obtained in 10 iterations.

ans =

no answer
```

2.1.2 Newton's Method for Solving Equations f(x) = 0

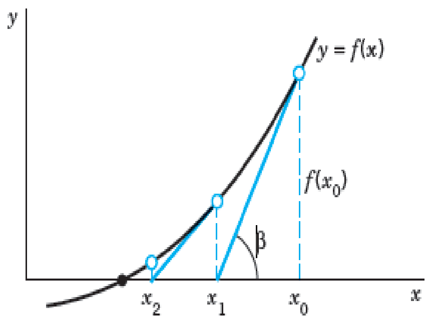


Fig. 3: Newton's method

Newton's method, also known as Newton-Rapson's method, is an iteration method for solving equations $f(x) = 0$, where f is assumed to have a continuous derivative f'. The method is commonly used because of its simplicity and great speed.

$$\tan\beta = f'(x_0) = \frac{f(x_0)}{x_0 - x_1}$$

$$x_0 - x_1 = \frac{f(x_0)}{f'(x_0)}, \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Table 1: Newton's Method for Solving Equations f(x)=0

ALGORITHM NEWTON$(f, f', x_0, \epsilon, N)$ This algorithm computes a solution of $f(x) = 0$ given an initial approximation x_0. INPUT: $f, f', initial\ approximation\ x_0, tolerance\ \epsilon > 0, maximum\ number\ of\ iterations\ N$ OUTPUT: Approximate solution $x_n (n \leq N)\ or\ message\ failure$ For n=0,1, 2, …,N-1 do: Compute $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ If $\lvert x_{n+1} - x_n \rvert \leq \epsilon \lvert x_{n+1} \rvert$ then OUTPUT x_{n+1}. Stop. [Procedure completed successfully] End OUTPUT "Failure". Stop. [Procedure completed unsuccessfully after N iterations] End NEWTON

Example: Apply Newton's method to the equation

$$f(x) = x^3 + x - 1 = 0.$$

Solution: We have

$$x_{n+1} = x_n - \frac{x_n{}^3 + x_n - 1}{3x_n{}^2 + 1} = \frac{2x_n{}^3 + 1}{3x_n{}^2 + 1}$$

$$Starting\ from\ x_0 = 1, we\ obtain$$

$$x_1 = 0.750000, x_2 = 0.686047,$$

$$x_3 = 0.682340, x_4 = 0.682328, \ldots$$

MATLAB Function of Newton's Method

```
Editor - C:\Users\CUBMW\Marlarwinkhin\matlabprogram\newton.m
newton.m
function x = newton (f,f1,x0 ,n,tol)
for i = 1:n
    xi= x0 - f(x0)/ f1(x0);
    if abs(xi-x0)<= tol*abs(xi)
        break
    end
        x0=xi;
    fprintf('x(%d)=%9.6f\n',i,xi);

end
if i<n
    fprintf('Ans=%9.6f\n',xi);
elseif i==n
    fprintf('Solution was not obtained in %i iterations.\n',n)
    x=('no answer');
end
```

Fig. 4: MATLAB Function of Newton's Method

```
>> f=@(x)x^3+x-1;f1=@(x)3*x^2+1;
>> x0=1;tol=0.000001;n=10;
>> newton(f, f1, x0, n, tol)
x(1)= 0.750000
x(2)= 0.686047
x(3)= 0.682340
x(4)= 0.682328
Ans= 0.682328
```

2.1.3 Secant Method for Solving f(x) = 0

The secant method is a variation on the theme of Newton's method. It uses a succession of roots of secant lines to better approximate a root of a function f.

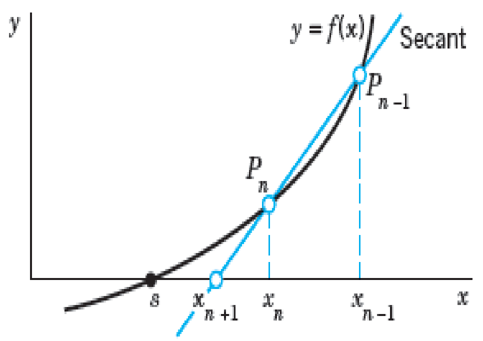


Fig. 5: Secant method

$$x_{n+1} = \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}$$

MATLAB Function of Secant Method

```
Editor - C:\Users\CUBMW\Marlarwinkhin\matlabprogram\secant.m
secant.m
function x = secant (f,x0,x1 ,n,tol)
for i = 2:n
    xi= (x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
    if abs(xi-x1)<= tol*abs(xi)
        break
    end
        x0=x1;
        x1=xi;
    fprintf('x(%d)=%9.6f\n',i,xi);

end
if i<n
    fprintf('Ans=%9.6f\n',xi);
elseif i==n
    fprintf('Solution was not obtained in %i iterations.\n',n)
    x=('no answer');
end
```

Fig. 6: MATLAB Function of Secant Method

```
>> f=@(x)x^3+x-1;f1=@(x)3*x^2+1;
>> x0=1;x1=0;tol=0.000001;n=10;
>> secant(f, x0, x1, n, tol)
x(2)= 0.500000
x(3)= 0.800000
x(4)= 0.663755
x(5)= 0.680532
x(6)= 0.682357
x(7)= 0.682328
Ans= 0.682328
```

2.2 Linear Systems: Solution by Iteration: Gauss-Seidel Iteration Method

This is an iterative method used to solve a linear system of equations. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either diagonally dominant (i.e. the magnitude of the diagonal entry in a row is larger than or equal to the sum of magnitudes of all the others entries in that row), or symmetric and positive definite.

Example: Gauss-Seidel Iteration

We consider the linear system

$$\text{(1)}\qquad \begin{aligned} x_1 - 0.25x_2 - 0.25x_3 &= 50 \\ -0.25x_1 + x_2 - 0.25x_4 &= 50 \\ -0.25x_1 + x_3 - 0.25x_4 &= 25 \\ - 0.25x_2 - 0.25x_3 + x_4 &= 25. \end{aligned}$$

We write the system in the form,

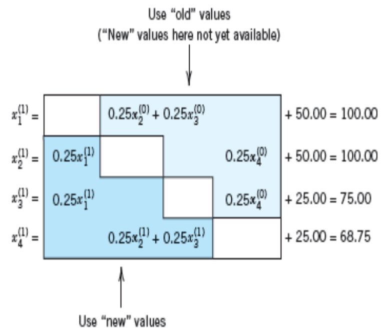
$$\text{(2)}\qquad \begin{aligned} x_1 &= 0.25x_2 + 0.25x_3 + 50 \\ x_2 &= 0.25x_1 + 0.25x_4 + 50 \\ x_3 &= 0.25x_1 + 0.25x_4 + 25 \\ x_4 &= 0.25x_2 + 0.25x_3 + 25. \end{aligned}$$

These equations are now used for iteration; that is, we start from a (possibly poor) approximation to the solution, say

$$x_1^{(0)} = 100, x_2^{(0)} = 100, x_3^{(0)} = 100, x_4^{(0)} = 100,$$

And compute from the above equations a perhaps a better approximation

(3)



These equations (3) are obtained from (2) by substituting on the right the most recent approximation for each unknown. In fact, corresponding values replace previous ones as soon as they have been computed, so that in the second and third equations we use $x_1^{(1)} (not\ x_1^{(0)})$, and in the last equation of (3), we use $x_2^{(1)}\ and\ x_3^{(1)} (not\ x_2^{(0)}\ and\ x_3^{(0)})$. Using the same principle, we obtain the next step

$$
\begin{aligned}
x_1^{(2)} &= \phantom{0.25x_1^{(2)}} \quad 0.25x_2^{(1)} + 0.25x_3^{(1)} \qquad + 50.00 = 93.750\\
x_2^{(2)} &= 0.25x_1^{(2)} \qquad\qquad\qquad + 0.25x_4^{(1)} + 50.00 = 90.625\\
x_3^{(2)} &= 0.25x_1^{(2)} \qquad\qquad\qquad + 0.25x_4^{(1)} + 25.00 = 65.625\\
x_4^{(2)} &= \phantom{0.25x_1^{(2)}} \quad 0.25x_2^{(2)} + 0.25x_3^{(2)} \qquad + 25.00 = 64.062
\end{aligned}
$$

Further steps give the values

x_1	x_2	x_3	x_4
89.062	88.281	63.281	62.891
87.891	87.695	62.695	62.598
87.598	87.549	62.549	62.524
87.524	87.512	62.512	62.506
87.506	87.503	62.503	62.502

Hence convergence to the exact solution

$$x_1 = x_2 = 87.5, x_3 = x_4 = 62.5$$

Table 2: Gauss-Seidel Iteration

ALGORITHM GAUSS-SEIDEL$(a, b, x^{(0)}, N)$
This algorithm computes a solution x of the system Ax=b given an initial approximation $x^{(0)}$, where
$A = [a_{jk}]\ is\ an\ n \times n\ matrix\ with\ a_{jj} \neq 0, j = 1,2, \ldots n.$
INPUT :
$a, b, initial\ approximation\ x^{(0)}, maximation\ iterations\ N$
OUTPUT: Approximate solution $x^{(N)} = [x_j^{(N)}]$
For m=0,…,N-1 do:
For j=1,…,n do:
$x_j^{(m+1)} = \frac{1}{a_{jj}}(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(m+1)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(m)}$
End
End
OUTPUT: Approximate solution $x^{(N)} = [x_j^{(N)}]$
End GAUSS-SEIDEL

MATLAB Function of Gauss-Seidel Iteration

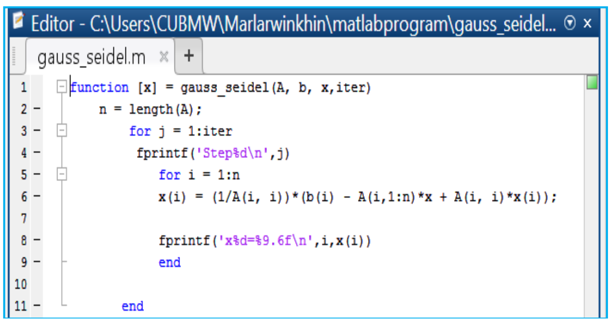


Fig. 7: MATLAB Function of Gauss-Seidel Iteration

```
>> A=[1 -0.25 -0.25 0;-0.25 1 0 -0.25;-0.25 0 1 -0.25;0 -0.25 -0.25 1];
>> b=[50;50;25;25];
>> x=[100;100;100;100]; iter=5;
>> gauss_seidel(A, b, x, iter)
Step1
x1=100.000000
x2=100.000000
x3=75.000000
x4=68.750000
Step2
x1=93.750000
x2=90.625000
x3=65.625000
x4=64.062500
Step3
x1=89.062500
x2=88.281250
x3=63.281250
x4=62.890625
Step4
x1=87.890625
x2=87.695313
x3=62.695313
x4=62.597656
Step5
x1=87.597656
x2=87.548828
x3=62.548828
x4=62.524414

ans =

   87.5977
   87.5488
   62.5488
   62.5244
```

2.3 Least Squares Method

Our method of curve fitting can be generalized from a polynomial of degree m

$$p(x) = b_0 + b_1x + b_2x^2 + \cdots + b_mx^m$$

In the case of a quadratic polynomial

$$p(x) = b_0 + b_1x + b_2x^2$$

The normal equations are (summation from 1 to n)

$$b_0n + b_1 \sum x_j + b_2 \sum x_j{}^2 = \sum y_j$$

$$b_0 \sum x_j + b_1 \sum x_j{}^2 + b_2 \sum x_j{}^3 = \sum x_jy_j$$

$$b_0 \sum x_j{}^2 + b_1 \sum x_j{}^3 + b_2 \sum x_j{}^4 = \sum x_j{}^2y_j$$

Example: Fit a straight line and a parabola through the data (0, 1.8), (1, 1.6), (2, 1.1), (3, 1.5), (4, 2.3).

Solution:
For the normal equations, we need n=5,

$$\sum x_j = 10 \sum x_j{}^2 = 30, \sum x_j{}^3 = 100, \sum x_j{}^4 = 354,$$
$$\sum y_j = 8.3, \sum x_jy_j = 17.5, \sum x_j{}^2y_j = 56.3.$$

For a straight line, the normal equations are

$$5a + 10b = 8.3$$
$$10a + 30b = 17.5$$

The solution is a= 1.48 and b= 0.09

We obtain the straight line

$$y = 1.48 + 0.09x.$$

For a parabola, these equations are

$$5b_0 + 10b_1 + 30b_2 = 8.3$$
$$10b_0 + 30b_1 + 100b_2 = 17.5$$
$$30b_0 + 100b_1 + 354b_2 = 56.3$$

Solving them we obtain the quadratic least squares parabola

$$y = 1.8943 - 0.7386x + 0.2071x^2$$

MATLAB Function of Least Squares Method

Editor - C:\Users\CUBMW\Marlarwinkhin\matlabprogram\leastsquared.m

forward_interpolation.m | gauss_seidel.m | leastsquared.m

```
function [C]=leastsquared(x,y,m)
n=length(x);F=zeros(n:m+1);
%Fill the columns of F with the powers of x
for k=1:m+1
    F(:,k)=x'.^(k-1);
end
A=F'*F
b=F'*y'
C=A\b;
t = min(x):0.5:max(x);n = length(t);
[f]=[];
for i = 1:n
       f(i) = C(m+1);
   for j = m:-1:1
      f(i) = C(j) + f(i)*t(i);
   end
end
%data visualization
figure
plot(t,f)   %the least_squares polynomial
hold on
plot(x, y, 'r*')    %the data points
 title('This is a figure of equation by using Least Squared Method');
 xlabel('x-axis')
 ylabel('y-axis')
grid on
```

Fig. 8: MATLAB Function of Least Squares Method

```
>> x=[0 1 2 3 4];y=[1.8 1.6 1.1 1.5 2.3];m=1;
>> leastsquared(x, y, m)

A =

     5    10
    10    30


b =

    8.3000
   17.5000


ans =

    1.4800
    0.0900
```

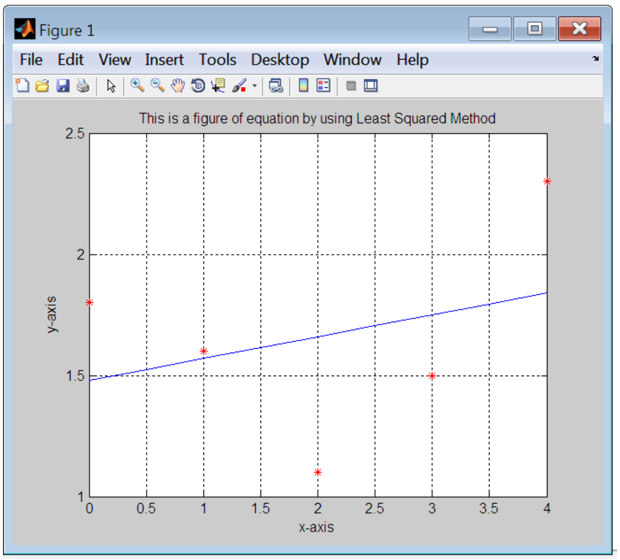


Fig. 9: Least Squares Straight Line

```
>> x=[0 1 2 3 4];y=[1.8 1.6 1.1 1.5 2.3];m=2;
>> leastsquared(x, y, m)

A =

     5    10    30
    10    30   100
    30   100   354


b =

    8.3000
   17.5000
   56.3000


ans =

    1.8943
   -0.7386
    0.2071
```

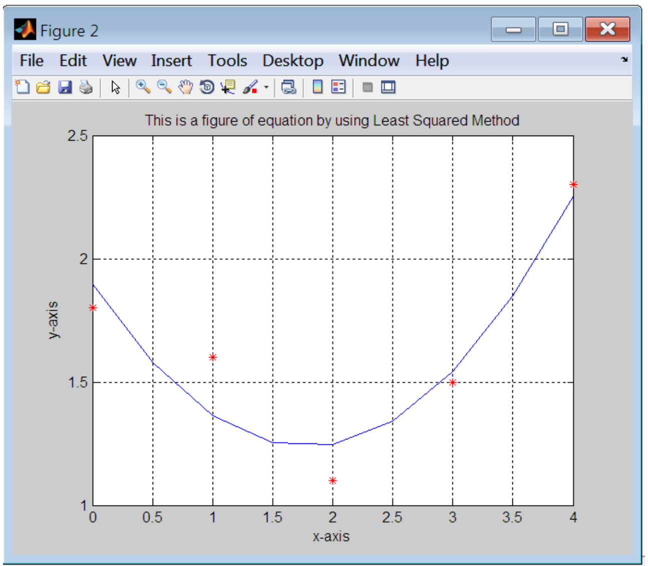


Fig. 10: Least Squares Parabola

3. RELATED WORKS

Computer Algebra Systems (CASs) are software programs with the ability to carry out mathematical computations both numerically and algebraically. Examples of popular CASs include MATLAB, Maple and Mathematica. CASs are increasingly being used in teaching and learning mathematics at all levels of the education system, including at the secondary and university level. [1]

An urgent need for integrating MATLAB as a didactical tool for calculus at the university was necessary. Currently, the teaching and learning of mathematics at the University are mostly traditional. The decision of incorporating MATLAB has been made primarily in an attempt to motivate students and help them to develop the necessary mathematical concepts and skills needed to succeed in their engineering majors. [2]

Today the researchers concentrate on the latest innovations in mathematics teaching. The main constraints in teaching the numerical course by using MATLAB are lack of computer knowledge, lack of programming knowledge, lack of interest to learn MATLAB coding and to apply in solving the real-life problems using numerical techniques, etc. [3]

4. CONCLUSIONS

In this paper, the utilization of MATLAB software in the teaching of the numeric problems such as Fixed-Point Iteration, Newton's Method, Secant Method, Gauss-Seidel Iteration and

Least Squared Method are demonstrated. MATLAB software usage is intended to improve the understanding of numeric problems. The numerical analysis must be taught in such a way so that the students will able to write computer programs to solve numerical problems with MATLAB depending upon the nature of problems. And they will be able to perform both hand computation and programming applied in MATLAB.

5. REFERENCES

[1] A. Nyamapfene, "Integrating MATLAB into First-Year Engineering Mathematics ". 2016 IEEE International Conference on Teaching and Learning in Education (ICTLE'16), Volume: 2016.

[2] M. Abdul Majid, Z. A. Huneiti, M. A. Al-Naafa, W. Balachandran, "A Study of the Effects of Using MATLAB as a Pedagogical Tool for Engineering Mathematics Students". iJOE- Volume 9, Issue 2, May 2013, pp-27-35.

[3] M. Haque Bhuyan, S. Shermin Azmiri Khan, "Teaching Numerical Analysis Course for Electrical Engineering Students using MATLAB". SEU Journal of Science and Engineering, Vol. 10, No. 2, December 2016, pp-38-46.

[4] S. I. Barry, T. Webb, "Multi-disciplinary approach to teaching numerical methods to engineers using MATLAB".ANZIAM J. 47 (EMAC2005) pp-C216-C230, 2006.

[5] V. E. Martinez Luaces and G. E. Guineo Cobs. Numerical Calculus and Analytical Chemistry. Proc. 2nd Int. Conf. on Teaching Mathematics, Crete, Wiley, 2002.

[6] D. Smith. How people learn ... Mathematics. Proc. 2nd Int. Conf. on Teaching Mathematics, Crete, Wiley, 2002.

BIOGRAPHY

Marlar Win Khin
Lecturer
University of Computer Studies, Banmaw, Myanmar