Scilab Software as an Alternative Low-Cost Computing in Solving the Linear Equations Problem

Fahrul Agus[1*] and Haviluddin[1]

[1]*Faculty of Computer Science and Information Technology, Mulawarman University, Indonesia*

[*)]Corresponding author: fahrulagus@unmul.ac.id

Abstract. Numerical computation packages are widely used both in teaching and research. These packages consist of license (proprietary) and open source software (non-proprietary). One of the reasons to use the package is a complexity of mathematics function (i.e., linear problems). Also, number of variables in a linear or non-linear function has been increased. The aim of this paper was to reflect on key aspects related to the method, didactics and creative praxis in the teaching of linear equations in higher education. If implemented, it could be contribute to a better learning in mathematics area (i.e., solving simultaneous linear equations) that essential for future engineers. The focus of this study was to introduce an additional numerical computation package of Scilab as an alternative low-cost computing programming. In this paper, Scilab software was proposed some activities that related to the mathematical models. In this experiment, four numerical methods such as Gaussian Elimination, Gauss-Jordan, Inverse Matrix, and Lower-Upper Decomposition (LU) have been implemented. The results of this study showed that a routine or procedure in numerical methods have been created and explored by using Scilab procedures. Then, the routine of numerical method that could be as a teaching material course has exploited.

INTRODUCTION

There are some problems in the transfer of knowledge in teaching numerical methods and modeling simulation fields, such as the elimination process line at the solving simultaneous linear equations. Many students who have failed in the theoretical understanding of classical algebra and monotonous, for example, the difficulty of implementing an algebraic formula into a computer programming language. In addition, the analytic process manually on complex evidentiary issues have limitations. Therefore, innovation and variation in this learning by doing numerical computation process are indispensable.

In general, analytic mathematical problem can be solved by numerical processes, such as linear and non-linear equations, calculus derivative and integral, differential equations, as well as series and error. Meanwhile, the search of linear equations can be solved by various methods, including Gaussian elimination, Gauss-Jordan, matrix inverse, decomposition Lower-Upper, and Jacobi and Gauss-Seidel and so forth.

Nowadays, the complexity of a linear problem is getting complicated. There are multivariable in a linear/non-linear functions. For this situation, manually processing is no longer possible. Therefore, computational processing is required. There are a lot of computational software for the completion of linear solutions, including MS. Excel, LINDO, and MATLAB, so on. However, these software are commercial or proprietary, so to use legally, users must purchase a license [1,2].

However, there are many alternatives that free software and open source. The software is also able to perform numerical computation process with a high level of accuracy too. In this paper, we proposed as alternative Scilab software in solving complex linear problems. The main reason is that the Scilab software can be developed in a free, open and non-commercial and inexpensive [3-5].

The rest of this paper is organized as following; recent related work is discussed and presented in section 2. In section 3, is proposed the numerical method. In section 4 a set of experiments is presented numerical methods. Finally, conclusion and future work.

The 1st International Conference on Mathematics, Science, and Computer Science (ICMSC) 2016
AIP Conf. Proc. 1813, 040005-1–040005-9; doi: 10.1063/1.4975970
Published by AIP Publishing. 978-0-7354-1481-5/$30.00

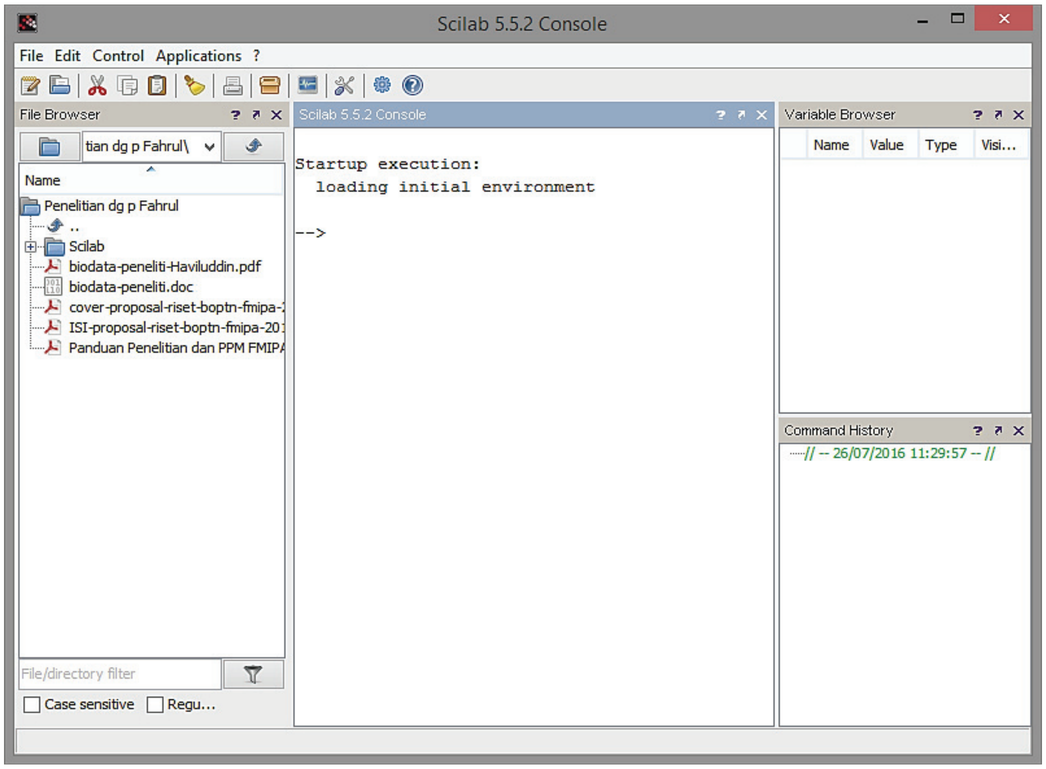
Related Works

Several researcher have been explored Scilab as a tools in artificial intelligence (AI) area, for example [6] have been explored Scilab for time series forecasting using ANN (i.e., ANN-GD using extended back propagation (EBP) algorithm, ANN-GA using genetic algorithm (GA), and ANN-DE using differential evolution (DE)). The datasets were analysis two time series, from the well-known Hyndman's time series data library exported from January 1961 to October 1975 and monthly Wisconsin employment time series from January 1962 to December 1975. The results showed that Scilab was an alternative tool in forecasting problem that easy to implemented and inexpensive. Then, [3] have been explored the Scilab for linear algebra for teaching activity. The experimental results revealed that Scilab commands are fairly simple in write mathematic formula, especially in linear algebra. Moreover, students said that this facilitates makes a class more dynamic and interesting. Then, [7] have been used Scilab to improve a convergence speed of back propagation neural network (BPNN). The convergence speed is a crucial parameter for good training. The Wisconsin Breast Cancer (UCI) Database of 699 patients was used for experiment. The results indicated that Scilab was evolving swiftly and good software related to costly. The system was working well. The Proposed method of weight updation was providing faster convergence while keeping the same accuracy. Later, [8] have been used the Scilab 5.4.0 Programming to solve mathematical model (i.e., Ordinary Differential Equations (ODE) with modify Euler's method called Harmonic Euler). The results revealed that Scilab can be used as an alternative software programming especially to solve numerical method. Later, [9] have been conducted novel multilevel membraneless enzymatic biofuel cell with wireless sensor. The experimental data chemicals were obtained from Sigma-Aldrich and used as received without further purification. The results indicated that Scilab could be used as tools for block diagram model environment, especially in 3D architecture of a membraneless enzymatic biofuel cell.

METHODOLOGY

Introducing to Scilab Programming

Scilab is a programming language associated with a rich collection of numerical algorithms covering many aspects of scientific computing problems. Scilab was written by *Institut Nationale de Recherche en Informatique et en Automatique – INRIA* (National Institute for Informatics and Automation Research) the French National Research Institution, in 1990. Scilab is free numerical computational package that have many of the same features with proprietary software (i.e., MATLAB, Minitab, etc.). The Scilab is available to download on the Linux, Windows, and Mac OS X operating systems. To download Scilab go to web page at www.scilab.org/products/scilab/download. The Scilab also interface with LabVIEW, a platform and development environment for visual programming languages. Furthermore, Scilab is free source software and no need to pay for license because it is provided under the CeCILL license. The Scilab also provided an editor to edit script easily, namely Scinotes. The editor could be accessed from the menu of the console, under the *Application> Editor* menu or from the console [4].

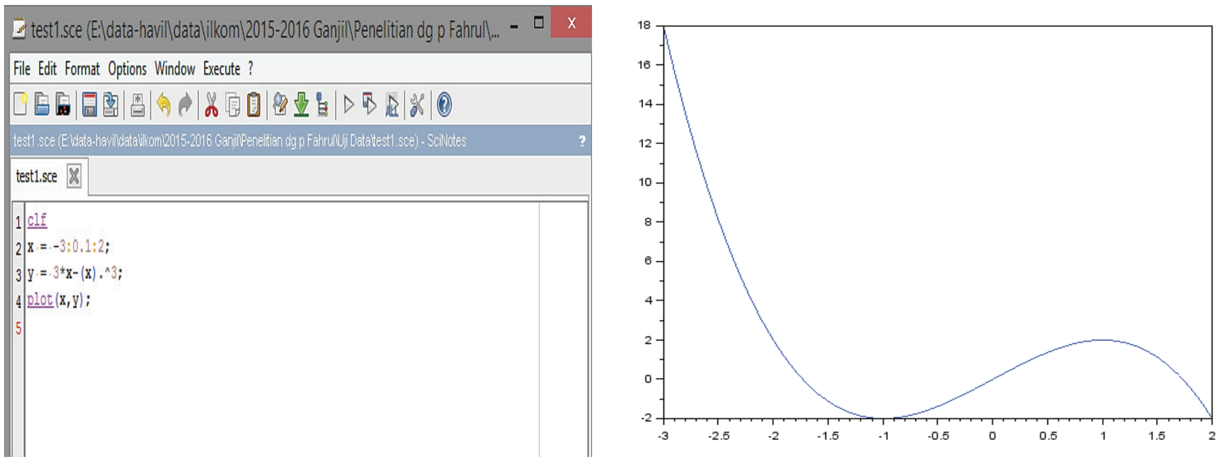


```
Startup execution:
  loading initial environment

-->
```

```
// -- 26/07/2016 11:29:57 -- //
```

FIGURE 1. The Scilab Console


test1.sce (E:\data-havil\data\ilkom\2015-2016 Ganjil\Penelitian dg p Fahrul\...
File Edit Format Options Window Execute ?
test1.sce (E:\data-havil\data\ilkom\2015-2016 Ganjil\Penelitian dg p Fahrul\Uji Data\test1.sce) - SciNotes
test1.sce
18
16
14
12
10
8
6
4
2
0
-2
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

```
clf
x = -3:0.1:2;
y = 3*x-(x).^3;
plot(x,y);
```

FIGURE 2. The Scilab Text Editor (Scinotes)

Numerical Methods

Numerical methods are a topics related to generate the matrices to solving the linear equation, ordinary differential equation (ODE's) and numerical integration (NI). Numerical methods is use mathematical modeling forms [10,11]. In this study, four numerical methods such as Gaussian Elimination, Gauss-Jordan, Inverse Matrix, and Lower-Upper Decomposition (LU) are briefly discussed.

Gaussian Elimination Method

Gaussian elimination method is universally known as method for solving simultaneous linear equations. This method was proposed by Karl Friedrich Gauss. The Gaussian elimination is an operating process values in a matrix, so that matrix becomes more modest. The process is to perform the operation of the line, so that the matrix into echelon-line form. This method can also be used to solve linear equations using matrix. The linear equations is modified by inserting into the matrix augmentation and operate it [3,12].

Gauss-Jordan Method

Elimination Gauss-Jordan method is an improvement of a Gaussian elimination method. The results is much better than Gaussian elimination. The trick is to continue the operation of the line of Gaussian elimination to produce a matrix that reduced row echelon form [3,12].

Inverse Matrix Method

Inverse matrix method could be also be used to compute for solving linear equations. However, this method is less effective than Gaussian elimination method, because more of the computing process is needed. Nevertheless, this method is effective, if a similar matrix with different vectors [12]. According to Matlab-Mathworks "The inverse is returned as a matrix of the same type as the input matrix. If the matrix is not invertible, then *fail* is returned. If the input does not evaluate to a matrix, then a symbolic call of inverse is returned" [13].

Lower-Upper Decomposition (LU) Method

Lower-Upper Decomposition (LU) method is also a modification of Gaussian elimination method, because some of its steps that must be discarded, but then must be used in the process of LU decomposition method [12]. According to Matlab-Mathworks "The LU function expresses a matrix A as the product of two essentially triangular matrices, one of them a permutation of a lower triangular matrix and the other an upper triangular matrix. The factorization is often called the *LU*, or sometimes the *LR*, factorization. A can be rectangular" [13].

RESULTS AND DISCUSSIONS

In this experiment, the common topics that involved in engineering solving and basic conceptual using Scilab programming was implemented. The four numerical methods such as Gaussian Elimination, Gauss-Jordan, Inverse Matrix, and Lower-Upper Decomposition (LU) were used as a case study.

The matrix *A* has *m* rows and *n* columns. Scilab has multidimensional matrices. This paper introduce the operations of create a function, generating and using matrices, and applying to solve problem of four numerical methods.

Case Study 1

Create a function for Gauss Method

```
function x=backward(a,b,n)
  x(n)=b(n)/a(n,n)
  for k=n-1:-1:1
    sigma=0
    for j=k+1:n
      sigma=sigma+a(k,j)*x(j);
      disp([j,k,sigma])
    end
    x(k)=[b(k)-sigma]/a(k,k)
    disp(x)
  end
endfunction
```

Generating and using matrices

```
a=[4 -1 2 3;0 -2 7 -4;0 0 6 5;0 0 0 3];
b=[20;-7;4;6];
```

Solution

```
backward(a,b,4);
y =linsolve(a,-b);
disp(y);
```

Case Study 2
Create a function for Gauss Method

```
function Gauss(A,B)
n = length(B);
Aug = [A,B];
Matrix = Aug(:,:); disp(Matrix ," Initial Matrix = ")
// Forward Elimination
for j = 1:n-1
  // Partial Pivoting
  [zero,t] = max(abs(Aug(j:n,j)));
  lrow = t(1)+j-1;
  Aug([j,lrow],:) = Aug([lrow ,j],:);
    for i = j+1:n
      Aug(i,j:n+1) = Aug(i,j:n+1) - Aug(i,j) / Aug (j,j) * Aug(j,j:n+1);
    end
  end
  Matrix = Aug(:,:); disp(Matrix ," Triangular Matrix = ")
  // Backward Substitution
  x = zeros(n,1);
  x(n) = Aug(n,n+1) / Aug(n,n);
  for i = n-1:-1:1
    x(i) = (Aug(i,n+1)-Aug(i,i+1:n)*x(i+1:n))/Aug(i, i);
  end
  disp(" Results x, y, and z, w = "); disp(strcat(["x = ",string(x(1))])); disp(strcat(["y = ",string(x(2))]));
disp(strcat(["z = ",string(x(3))])); disp(strcat(["w = ",string(x(4))]))
endfunction
```

Solution

```
a=[4 -1 2 3;0 -2 7 -4;0 0 6 5;0 0 0 3];
b=[20;-7;4;6];
Gauss(a,b)
Initial Matrix=                          Triangular Matrix =
        4.  - 1.   2.   3.   20.           4.  - 1.   2.   3.   20.
        0.  - 2.   7.  - 4.  - 7.          0.  - 2.   7.  - 4.  - 7.
        0.   0.   6.   5.   4.             0.   0.   6.   5.   4.
        0.   0.   0.   3.   6.             0.   0.   0.   3.   6.
Results x, y, and z, w =
x = 3   y = -4    z = -1    w = 2
```

Case Study 3
Create a function for Gauss-Jordan Method

```
function Jordan(A,B)
n = length (B);
Aug = [A,B];
Matrix = Aug(:,:);
disp(Matrix ," Initial Matrix = ")
//Forward Elimination
    for j = 1:n-1
        for i = j+1:n
```

```
            Aug(i,j:n+1) = Aug(i,j:n+1) - Aug(i,j) / Aug (j,j) * Aug(j,j:n+1);
          end
      end
Matrix = Aug(:,:); disp(Matrix ," Triangular Matrix = ")
// Backward Elimination
      for j = n:-1:2
        Aug(1:j-1,:) = Aug(1:j-1,:) - Aug(1:j-1,j) / Aug (j,j) * Aug(j,:);
      end
Matrix = Aug(:,:); disp(Matrix ," Diagonal Matrix = ")
// Diagonal Normalization
for j=1:n
      Aug(j,:) = Aug(j,:) / Aug(j,j);
end
Matrix = Aug(:,:); disp(Matrix ," Matrix Identity = ")
x = Aug(:,n+1);
disp(" Results x, y, z and w = "); disp(strcat(["x = ",string(x(1))])); disp(strcat(["y = ",string(x(2))])); disp(strcat(["z
= ",string(x(3))])); disp(strcat(["w = ",string(x(4))]))
endfunction
```

Solution

```
a                                             b
      a  =                                     b  =
         4.  - 1.    2.    3.                     20.
         0.  - 2.    7.  - 4.                   - 7.
         0.    0.    6.    5.                     4.
         0.    0.    0.    3.                     6.
Jordan(a,b)
Initial Matrix =                              Triangular Matrix =
         4.  - 1.    2.    3.    20.             4.  - 1.    2.    3.    20.
         0.  - 2.    7.  - 4.  - 7.              0.  - 2.    7.  - 4.  - 7.
         0.    0.    6.    5.    4.              0.    0.    6.    5.    4.
         0.    0.    0.    3.    6.              0.    0.    0.    3.    6.
Diagonal Matrix =                               Matrix Identity =
         4.    0.    0.    0.    12.             1.    0.    0.    0.    3.
         0.  - 2.    0.    0.    8.              0.    1.    0.    0.  - 4.
         0.    0.    6.    0.  - 6.              0.    0.    1.    0.  - 1.
         0.    0.    0.    3.    6.              0.    0.    0.    1.    2.
Results x, y, z and w =
x = 3   y = -4   z = -1   w = 2
```

Case Study 4
Create a function for Invers Matrix Method

```
function Invers(A,B)
n = length (A(1,:));
Aug = [A,eye(n,n)];
Matrix = Aug(:,:); disp(Matrix ," Initial Matrix = ")
// Forward Elimination
      for j = 1:n-1
        for i = j+1:n
         Aug(i,j:2*n) = Aug(i,j:2*n) - Aug(i,j) / Aug(j,j) * Aug(j,j:2*n);
        end
      end
      Matrix = Aug(:,:); disp(Matrix ," Triangular Matrix = ")
      // Backward Elimination
      for j = n:-1:2
```

```
    Aug(1:j-1,:) = Aug(1:j-1,:) - Aug(1:j-1,j) / Aug (j,j) * Aug(j,:);
   end
   Matrix = Aug(:,:); disp(Matrix ," Diagonal Matrix = ")
   // Diagonal Normalization
   for j=1:n
      Aug(j,:) = Aug(j,:) / Aug(j,j);
   end
   Matrix=Aug(:,:); disp(Matrix,"Matrix Identity & Invers Results =")
   Inv_A = Aug(:,n+1:2*n);
   // Invers from A (A-1)
   disp(Inv_A ," Invers from A = ")
   x = Inv_A*B
   disp(" Results x, y, z and w = "); disp(strcat(["x = ",string(x(1))])); disp(strcat(["y = ",string(x(2))]));
disp(strcat(["z = ",string(x(3))])); disp(strcat(["w = ",string(x(4))]))
endfunction
```

Solution

```
a                                                   b
  a  =                                                b  =
    4.  - 1.    2.    3.                               20.
    0.  - 2.    7.  - 4.                              - 7.
    0.    0.    6.    5.                                4.
    0.    0.    0.    3.                                6.
Invers(a,b)
Initial Matrix =                                    Triangular Matrix =
    4. - 1.  2.   3.  1.  0.  0.  0.                    4. - 1.  2.   3.  1.  0.  0.  0.
    0. - 2.  7. - 4.  0.  1.  0.  0.                    0. - 2.  7. - 4.  0.  1.  0.  0.
    0.   0.  6.   5.  0.  0.  1.  0.                    0.   0.  6.   5.  0.  0.  1.  0.
    0.   0.  0.   3.  0.  0.  0.  1.                    0.   0.  0.   3.  0.  0.  0.  1.
Diagonal Matrix =
    4.    0.    0.    0.    1.  - 0.5    0.25          - 2.0833333
    0.  - 2.    0.    0.    0.    1.    - 1.1666667    3.2777778
    0.    0.    6.    0.    0.    0.     1.            - 1.6666667
    0.    0.    0.    3.    0.    0.     0.              1.
Matrix Identity & Invers Results =
    1.    0.    0.    0.    0.25  - 0.125   0.0625       - 0.5208333
    0.    1.    0.    0.    0.     - 0.5     0.5833333   - 1.6388889
    0.    0.    1.    0.    0.     0.        0.1666667   - 0.2777778
    0.    0.    0.    1.    0.     0.        0.            0.3333333
Invers from A =                                         Results x, y, z and w =
    0.25  - 0.125    0.0625       - 0.5208333            x = 3  y = -4  z = -1  w = 2
    0.    - 0.5      0.5833333  - 1.6388889
    0.      0.       0.1666667  - 0.2777778
    0.      0.       0.           0.3333333
```

Case Study 5

Create a function for Lower-Upper Decomposition (LU) Method

```
function Decomposition (A,B)
n = length (B);
L = zeros(n,n); // L = Initial Lower-Matrix
U = eye(n,n);   // U = Initial Upper-Matrix
// LU Decomposition
   for i = 1:n
      sum1 = zeros(n-i+1,1);
      for k = 1:i-1
```

```
            sum1 = sum1 + L(i:n,k) * U(k,i);
        end
        L(i:n,i) = A(i:n,i) - sum1;
        sum2 = zeros(1,n-i);
        for k = 1:i-1
            sum2 = sum2 + L(i,k) * U(k,i+1:n);
        end
        U(i,i+1:n) = (A(i,i+1:n) - sum2) / L(i,i);
    end
    Matrik = L(:,:); disp(Matrix ," Lower-Matrix = ");
    Matrik = U(:,:); disp(Matrix ," Upper-Matrix = ");
    // Forward Substitution
    D = ones(n,1);
    for i = 1:n
        sum3 = 0;
        for k = 1:i-1
            sum3 = sum3 + L(i,k) * D(k);
        end
        D(i) = (B(i) - sum3) / L(i,i);
    end
    // Back Substitution
    x = ones(n,1);
    for i = n:-1:1
        sum4 = 0;
        for k = i+1:n
            sum4 = sum4 + U(i,k) * x(k);
        end
        x(i) = D(i) - sum4;
    end
    disp(" Results x, y, z and w = ")
    disp(strcat(["x  =  ",string(x(1))]));  disp(strcat(["y  =  ",string(x(2))]));  disp(strcat(["z  =  ",string(x(3))]));
disp(strcat(["w = ",string(x(4))]))
endfunction
```

Solution

```
a                                              b
   a  =                                           b  =
     4.  - 1.   2.   3.                             20.
     0.  - 2.   7.  - 4.                           - 7.
     0.   0.   6.   5.                              4.
     0.   0.   0.   3.                              6.
Decomposition(a,b)
Lower-Matrix =                                 Upper-Matrix =
     4.   0.   0.   0.                              1. - 0.25   0.5   0.75
     0.  - 2.   0.   0.                             0.  1.    - 3.5   2.
     0.   0.   6.   0.                              0.  0.      1.   0.8333333
     0.   0.   0.   3.                              0.  0.      0.   1.
Results x, y, z and w =
x = 3   y = -4   z = -1   w = 2
```

CONCLUSION

This paper presented four numerical methods such as Gaussian Elimination, Gauss-Jordan, Inverse Matrix, and Lower-Upper Decomposition (LU). The process of numerical methods can be expressly and easily presented by using the Scilab as open source computational programming software in the market, especially in solving simultaneous linear equations. Means, several researchers have agreed that Scilab software is quite good and easy to use. In other words, Scilab is an alternative teaching and learning software in numerical method.

ACKNOWLEDGMENTS

This work has been supported by the BOPTN 2016 Faculty of Mathematics and Natural Science Universitas Mulawarman.

REFERENCES

1. Salleh Z., Yusop M.Y.M., Ismail S.B. Basic of Numerical Computational Using Scilab Programming. in the 2nd International Conference on Mathematical Applications in Engineering (ICMAE2012). pp. 1-8 (2012).
2. Baudin M., Couvert V. Optimization In Scilab. The Scilab Consortium - Digiteo / INRIA (2010).
3. Catarino P., Vasco P. Scilab in Linear Algebra. Applied Mathematical Sciences, vol. 8 2014, No. 28 pp. 1391-1399 (2014).
4. Baudin M., Steer S. Optimization with Scilab. present and future. pp. 1-8 (2009).
5. Wang Y., Liu L., Cao J., Feng K., Fu J., Li C. Random Forests Toolbox with Scilab and its Application. in The 7th International Conference on Computer Science & Education (ICCSE 2012) July 14-17 2012., Melbourne, Australia, pp. 1082-1085 (2012).
6. Panigrahi S., Karali Y., Behera H.S. Time Series Forecasting using Evolutionary Neural Network. Int J Comput Appl T (0975 – 8887), vol. 75– No.10, August 2013, pp. 13-17, (2013).
7. Singh A. Implementation of Back-Propagation Neural Network using Scilab and its Convergence Speed Improvement. Int. Journal of Electrical & Electronics Engg., vol. 2, Spl. Issue 1 (2015), pp. 192-194, (2015).
8. Yusop N.M.M., Hasan M.K., Rahmat M. Comparison New Algorithm Modified Euler in Ordinary Differential Equation Using Scilab Programming. Lecture Notes on Software Engineering. 3(3): 199-202, (2015).
9. Desmaële D., Renaud L., Tingry S. A wireless sensor powered by a flexible stack of membraneless enzymatic biofuel cells. Sensors and Actuators B, 220: 583–589 (2016).
10. Rao S. Applied Numerical Methods for Engineers and Scientist 3rd Edition (2002).
11. Salleh Z. Fundamental of Numerical Methods for Scientists and Engineers, 1st Edition (2011).
12. Urroz G.E. Matrix Operations with SCILAB. (2001).
13. MathWorks. LU Matrix Factorization. (2016).