International Journal of
Science and Engineering Investigations vol. 4, issue 46, November 2015
ISSN: 2251-8843

A Matlab Based Backward-forward Sweep Algorithm for Radial Distribution Network Power Flow Analysis

Kabir A. M.[1], Abubakar A. S.[2], Abdulrahman O.[3], Salisu S.[4]
[1,2,3,4] Department of Electrical and Computer Engineering, Ahmadu Bello University, Zaria, Nigeria
([1]abdulrashidmkabir@yahoo.com, [2]abubakaras@abu.edu.ng, [3]olaniyanabdulrahman@gmail.com, [4]salisu@live.com)

***Abstract*-** The continuous growth in electrical energy demand and pressing need for better quality of service has necessitated the need for continuous radial distribution network analysis. Power flow analysis forms the bases of network reconditioning for safety and reliability. In this paper, an improved Backward-Forward Sweep (BFS) technique for power flow analysis is achieved by breaking the conventional solution strategy into a number of logical steps (Matlab sub-functions) that simplifies the problem of node tracing and eliminates the need for sequential node numbering. The sub-functions are then integrated logically via a main-function called 'RADFLOW', forming a robust power flow algorithm. A block diagram of the proposed approach and the flow chart of the developed algorithm are presented. Finally, the effectiveness of the developed power flow Algorithm is demonstrated by performing power flow analysis using the 30, 33, and 69 bus standard IEEE test radial distribution networks. The plots of voltage profile and branch power flows of the selected test networks are presented followed by numerical values of the simulation results. The result shows that the proposed approach has a considerable fast rate of convergence and looks promising in solving power flow problems.

***Keywords*-** *Backward-Forward Sweep (BFS) Algorithm; Power Flow; Matlab; Distribution Network; Line Data; Bus Data*

I. INTRODUCTION

Power flow analysis is the determination of the steady state conditions of a power system for a set of specified power generations and load demand [1] . It involves the solution of a set of non-linear power flow equations. Applications, especially in the fields of power system optimization and distribution automation, require repeated fast power flow solutions. Due to the large number of interconnections and continuously increasing demand, the size and complexity of the present day power systems, have grown tremendously. In the last few decades efficient and reliable power flow techniques such as Gauss Seidel (GS), Newton-Raphson (NR) as shown in [2] and fast decoupled power flow (FDPF) as shown in [3] have been developed and widely used for powers system operation, control and planning. However, it has repeatedly been shown that these methods may become inefficient in the analysis of distribution systems due to the following facts [1]:

1. Distribution networks can be numerically ill-conditioned due to wide range of inherent radial structure;
2. Distribution power flow equations are different in nature from transmission power flow equation.

Consequently many power flow algorithms especially suited for distribution systems have been developed and well documented. These methods may be roughly categorized as node and branch based methods. The first category uses node voltages or current injections as state variables and requires information on the derivatives of network equations. The Z-bus method shown in [4], NR based algorithms as used in [5] and FDPF based algorithms shown in [6] belong to this category. The second category adopts branch currents or branch powers as state variables and involves only basic circuit laws. The backward/forward sweep based methods used in [7] and loop impedance as shown in [8] fall in this category. A fast decoupled G-matrix method for power flow (FDGPF) of distribution systems, based on equivalent current injections has been proposed in [9]. The method uses a constant and symmetric Jacobian matrix, which needs to be factorized only once.

However, the Jacobian matrix has been formed by omitting the reactance of the distribution lines with the assumption that (FDDPF) based on equivalent line current flows, which are rotated by appropriate line admittance angle for decoupling the problem, has been suggested in [6]. A compensation based technique that exploits radial structure to achieve high speed, robust convergence and low memory requirement, for weakly meshed distribution systems has been explained in [10].

A simple and efficient branch-to-node matrix based power flow (BNPF) for radial distribution systems has been presented in [11] [11] Effective radial distribution power flow based on improved Newton raphson's method had been integrated into Matlab GUIs [12] to simplify power flow computations. Power flow analysis and simulation Matlab toolbox had been developed for three phase power flow calculation [13] . Most of the existing algorithms, suggested in the literature, have been developed with an objective to reduce the computational burden by reducing the number of equations, approximating the Jacobian matrix, etc.

However, this paper presents an Improved FBS technique that is suitable for both large and small scale radial distribution network power flow analysis. The proposed approach is

achieved by breaking the conventional FBS technique into a number of distinct steps (sub-functions) which are programmed separately using Matlab. A Matlab function (called the main function or RADFLOW) that logically integrates these steps to form a flexible and robust power flow algorithm is then developed. The main function can easily be modified to suit application in areas like: Optimal Distributed Generation (DG) and Capacitor placement; Network reconfiguration etc. The proposed approach is used to perform power flow on three different standard test networks (30, 33, and 69 node) in order to demonstrate its effectiveness.

II. THE PROPOSED POWER FLOW MODEL

In the proposed power flow model, the network parameters are assumed to be stored in two matrices namely the line data and bus data matrices. The line data matrix is a matrix of size *(b x 5)*, where *b* is the number of branches in the network. The five columns in the line data matrix are defined as follows:

A. Column 1: Branch serial number;

B. Column 2: Sending end bus number (Frm_Bus);

C. Column 3: Receiving end bus number (To_Bus);

D. Column 4: Branch resistance;

E. Column 5: Branch reactance.

While, the bus data matrix is a matrix of size *(a x 5)*, where *a* is the number of buses (excluding the source bus) in the network. The five columns in the bus data matrix are defined as follows:

(a) Column 1: Bus serial number;

(b) Column 2: Magnitude of the load active power connected to a bus;

(c) Column 3: Power factor of the load connected to a bus;

(d) Column 4: Magnitude of the initial bus voltage;

(e) Column 5: initial bus voltage angle.

For simplicity, a ten nodes (excluding the source node) network as shown in Figure 1 is used in this paper for describing the proposed approach. In Figure 1, node '0' is the source node. In the proposed model, the source node is always represented by the index '0' so as to distinguish it from other network nodes, especially during power flow computations.

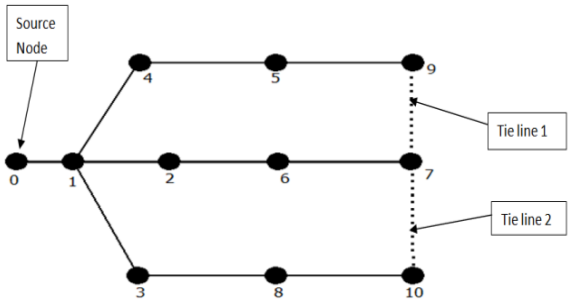


Figure 1. A Simple 10 Bus Radial Network with Two Tie-Switches (Lines/Branches)

The proposed method is governed by equations (1) to (5). The current injected into the source node (0) can be expressed using equation (1).

$$I = \left(\sum_{\substack{i=1 \\ i \neq s}}^{n} P_i + \sum_{\substack{j=1 \\ i \neq j}}^{n} P_{Loss.ij} + j\left(\sum_{\substack{i=1 \\ i \neq s}}^{n} Q_i + \sum_{\substack{j=1 \\ i \neq j}}^{n} Q_{Loss.ij}\right)\right) / V_s^* \quad (1)$$

Where:

$\sum_{\substack{i=1 \\ i \neq s}}^{n} P_i$ is the sum of the load real power connected to the entire receiving end buses;

$\sum_{\substack{i=1 \\ i \neq s}}^{n} Q_i$ is the sum of the load reactive power connected to the entire receiving end buses;

$\sum_{\substack{j=1 \\ i \neq j}}^{n} P_{Loss.ij}$ is the sum of the branch (ij) real power loss across the entire network branches;

$\sum_{\substack{j=1 \\ i \neq j}}^{n} Q_{Loss.ij}$ is the sum of the branch (ij) reactive power loss across the entire network branches;

V_s^* is the conjugate of the source voltage;

I is the current at the source end;

The real and reactive power loss across any branch *(ij)* can also be expressed using equations (2) and (3) respectively.

$$P_{Loss.ij} = I_{ij}^2 R_{ij} \quad (2)$$

$$Q_{Loss.ij} = I_{ij}^2 X_{ij} \quad (3)$$

where:

R_{ij} is the resistance of branch ij;

X_{ij} is the reactance of branch ij;

I_{ij} is the current flowing from bus i to j.

While the voltage drop across any branch *(ij)* in the network can also be expressed using equation (4).

$$V_{Drop.ij} = V_j - V_i = I_{ij} Z_{ij} \quad (4)$$

where the branch impedance (Z_{ij}) is a function branch resistance and reactance as shown in equation (5).

$$Z_{ij} = R_{ij} + jX_{ij} \quad (5)$$

where:

Z_{ij} is the impedance of branch ij;

$V_{Drop.ij}$ is the voltage drop across branch ij

V_j and V_i are the voltage at bus j and i respectively;

The characteristic of a network topology can be completely described using the first three columns of the line data. The first three columns of the line data of Figure 1 are shown in Table 1.

TABLE I. LINE DATA OF FIGURE 3

Line Number	From Bus	To Bus
1	0	1
2	1	2
3	1	3
4	1	4
5	4	5
6	2	6
7	6	7
8	3	8
9	5	9
10	8	10

Let: S_{inj} be a matrix of power injected into every bus, such that S_{inj} has a size of *n x1*; let *V* be a matrix of voltage at every bus, such that *V* has a size of *n x1*; let *SL* be a matrix of load power at every bus, such that *SL* has a size of *n x1*; and let *Z* be a matrix of line impedance, such that *Z* has a size of *n x1*, where *n* is the number of buses in the network, then *Z*, *SL*, and the initial value of *V*, can be derived from the line and bus data matrices using the following MATLAB scrip:

```
1  Z=line_data(:,4)+1i*linedata(:,5);
2  SL=bus_data(:,2).*bus_data(:,3)-...
          1i*bus_data(:,2).*sin(acos(bus_data(:,3)));
       3  V=bus_data(:,4).*(cos(bus_data(:,5))+...
                    1i*sin(bus_data(:,5)));
```

A. Information Matrix (IM)

The information matrix (IM) is a matrix of size (*b x Ls)*, where *b* is the number of branches in the network, and *Ls* is the number of branches on the longest section of the network (as traced from the source node (0)). Each row of the IM represents a possible path (backward trace). The IM is used during the backward sweep to trace each node back to the source node (0) and during the forward sweep to trace the source node (0) to each other node.. The following MATLAB Function (Script) can be used to build the information matrix using the line data matrix as input.

```
1       line_data(:,[2 3])=fliplr(line_data(:,[2 3]));
2       [v,w]=size(line_data);
3       bus=v;
4       x=zeros(bus);
5       x(:,[1 2])=line_data(:,[2 3]);
6       for i=1:bus
7           for j=2:bus
8               for k=1:bus
9                   if line_data(k,2)==x(i,j)
10                      x(i,[j j+1])=line_data(k,[2 3]);
11                  end
12              end
13          end
14      end
15      del=zeros (bus,1);
16      X=x;
17      for l=1:bus
18          if X(:,l)==del
19             [a,b]=size(x);
20             x(:,b)=[];
21          end
22      end
23      IM=sortc(x,1);
```

The IM of Fig.1 is shown in equation (6).

$$IM = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 5 & 4 & 1 & 0 \\ 6 & 2 & 1 & 0 \\ 7 & 6 & 2 & 1 \\ 8 & 3 & 1 & 0 \\ 9 & 5 & 4 & 1 \\ 10 & 8 & 3 & 1 \end{bmatrix} \quad (6)$$

B. Bus Incident Matrix

The bus incident matrix (BIM) is a square matrix of size (*b*), where *b* is the number of branches in the network. The BIM contains only ones and zeros. The BIM is used during the forward sweep process to estimate the branch voltage drop and power loss, and in the backward sweep process to estimate the power injected at every bus. IM is also used in building a BIM. BIM can be built using following MATLAB script:

```
[k,l]=size(IM);
bb=zeros(k)
for i=1:k
  for j=1:l
 if bbb(i,j)~=0
  for amk=1:k
      if bbb(amk,1)==bbb(i,j)
            gotit=amk;
    end
   end
    bb(i,gotit)=bbb(i,j);
  end
 end
end
BIM=bb;
[k,t]=size(BIM);
for q=1:k*t
    if BIM(q)~=0
      BIM(q)=1;
 end
end
```

The BIM of Figure 1 is shown in equation (7).

$$BIM = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (7)$$

C. Bus Injected Power (S_{inj})

In the proposed power flow model, the power injected into every bus is calculated during the backward sweep process. The load connected to each bus and the BIM are used in computing the power injected into every bus. The following MATLAB script can be used to estimate the power injected into every bus in the network (at the first backward sweep).

```
[k,t]=size(BIM);
II=[];
for e=1:t
   II=[II,SL];
end
II=II.*BIM;
II=sum(II);
II=II';
Sinj=II;
```

D. Power Loss (S_{loss})

In the proposed power flow model, the fist backward sweep is characterized by the assumption that the power loss on every line is equal to zero (0). At the subsequent backward sweep, the power loss (S_{loss}) on the lines is not equal to zero (0). Once the bus injected power (S_{inj}) is calculated, the power loss (S_{loss}) on the branches can be estimated using the following MATLAB script:

```
Sloss=(abs(Sinj-SL)./conj(V))^2).*Z;
```

Once the power loss (S_{loss}) is estimated, the bus injected power (S_{inj}) can be estimated during the subsequent backward sweep using the following MATLAB script:

```
[k,j]=size(IM);
kj=k*j;
for jk=1:kj
   for jj=1:k
      if line_data(jj,3)==IM(jk);
        total_loss(jk)=Sloss(jj);
      end
   end
end
Total_Loss=sum(total_loss,2);
Sinj=Sinj+Total_Loss;
```

E. Voltage Deviation (VD)

Once the branch power losses are estimated, the BIM and IM can then be used during the forward sweep (FS) to estimate the voltage deviation of the network buses. The following MATLAB script can be used to estimate the voltage drop at every bus.

```
vdrop=sqrt(Sloss./Z)*Z;
          [k,j]=size(BIM);
kj=k*j;
for jk=1:kj
   for jj=1:k
       if line_data(jj,3)==IM(jk);
        total_vdrop(jk)=vdrop(jj);
     end
end
end
VD=sum(total_drop,2);
```

Power flow simulation is performed using a number of MATLAB functions (Scripts) that work together to achieve power flow. These scripts are briefly described as follows:

i. RADFLOW: The main function that control the power flow execution. It requires the line data, bus data, and the source voltage as input. It has a MATLAB syntax of 'RADFLOW(line_data,bus_data,Vs)';

ii. Sortbus: A sub-function that performs the backward and forward sweep. It uses of the line data to sweep (add-up) a

desired parameter D across the entire network. It has a MATLAB syntax of 'sortbus(line_data,D)';

iii. Sortc: A sub-function that Sorts a matrix M in reference to a column C. It has a MATLAB syntax of 'sortc(M,C)';

iv. VDROP: A sub-function that computes the bus voltage deviation of a network using the line data. It has a MATLAB syntax of 'VDROP(line_data,drop', where drop is the voltage drop across the network branches.

v. Wizbus: A sub-function that constructs the information matrix (IM) using the line data. It has a MATLAB syntax of 'wizbus(line_data)';

Fig. 2 shows a block diagram of the power flow program in Matlab. In Fig. 2, the arrows indicate the directions of the data flow during the power flow simulation/ calculations.

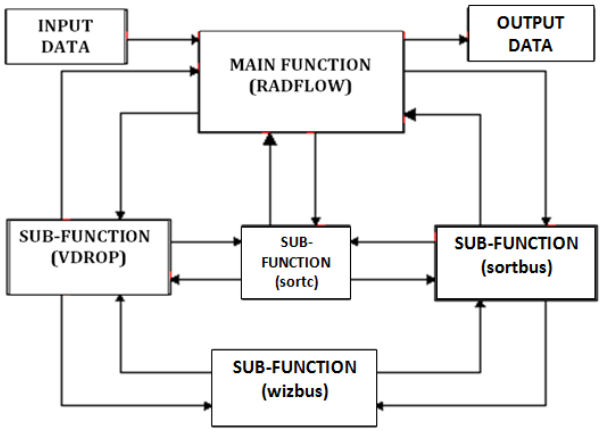


Figure 2. The Matlab Power Flow Function Block Diagram

III. The Proposed Power Flow Algorithm

In the proposed power flow algorithm, radial distribution power flow is achieved by logically executing the following set of instructions. A flow chart for the proposed algorithm is also illustrated in Fig. 3. The proposed power flow algorithm is as follows:

1. Construct the network's line data matrix, bus data matrix, and define the source voltage and the voltage error limit (e);
2. Execute the backward sweep, by logically tracing each and every node back to the source while computing the power flowing on the network branches and their respective branch power losses. This can be achieved using the Matlab function 'sortbus';
3. Execute the forward sweep, by logically tracing the source to every other node while computing the voltage drop along the branches (based on the results obtained from the second instruction (2) above). This can be achieved using the Matlab Function 'VDROP';
4. Update the node voltages using the results obtained from the third instruction (3) above, and store the updated node voltages as V_{F1};
5. Re-execute instructions two (2) and three (3) above; re-update the updated node voltages (as in four (4) above); and store the re-updated node voltages as V_{F2};
6. Compute the change in node voltage between two successive forward sweeps (which represents the absolute voltage error ($|V_{F1} - V_{F2}|$));
7. Repeat instructions two (2) to six (6), as long as the maximum absolute voltage error is greater than the error limit (e).
8. Print the results, if the condition in instruction seven (7) is violated.

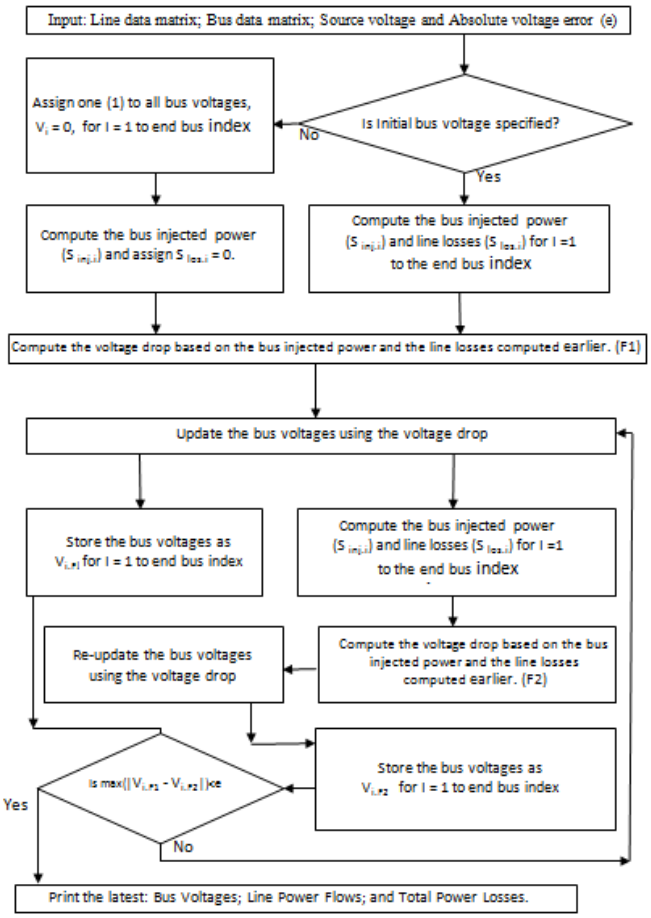


Figure 3. A Flow Chart of the Developed Power Flow Algorith33333

IV. Simulation Results

The proposed algorithm was developed into a Software package (Power Flow Simulator) using Matlab in order to aid power flow simulation. Simulation was carried out using an E-Machine PC with 2GB RAM/1.6GHz processor. The input to the power flow simulator includes the following:

1. Line data matrix;
2. Bus data matrix;
3. Source voltage (real and reactive component (pu));
4. Line and bus data types; and
5. Base value for voltage and power.

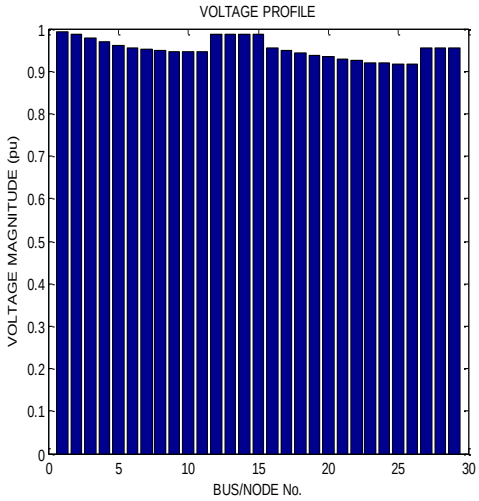
The output of the power flow simulator includes the following:

1. Branch active power (pu);
2. Bus/node voltage (pu);
3. Total power losses (pu);
4. Total simulation time (CPU sec.);
5. Total number of iterations. and
6. Plots of voltage profile and the branch power flow.

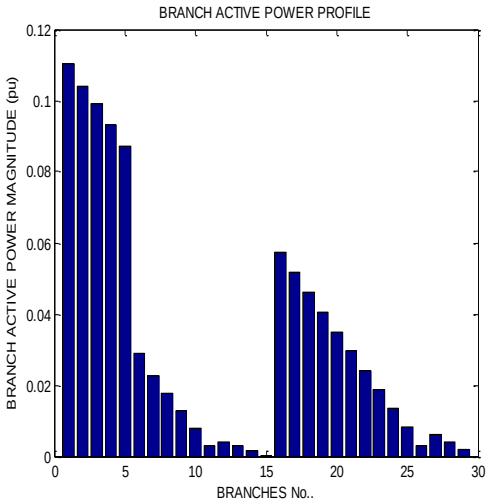
The entire simulation was carried out using the following settings:

Vs=1+0i;
Bus data type=type 1;
Line data type=type1;
Sbase=1; and
Vbase=1.

Fig. 4 (a) and (b) shows the bus voltage and branch power profiles of the 30 node network respectively; Fig. 5 (a) and (b) shows the bus voltage and branch power profiles of the 33 node network respectively; while Fig. 6 (a) and (b) shows the bus voltage and branch power profiles of the 69 node network respectively. Table 2 shows the values of node voltages of the 30, 33, and 69 node networks. Table 3 also compares the 30, 33, and 69 node networks in terms of total real power loss, simulation time, and number of iterations. The developed power flow simulator allows a maximum absolute voltage error of 1/11000.

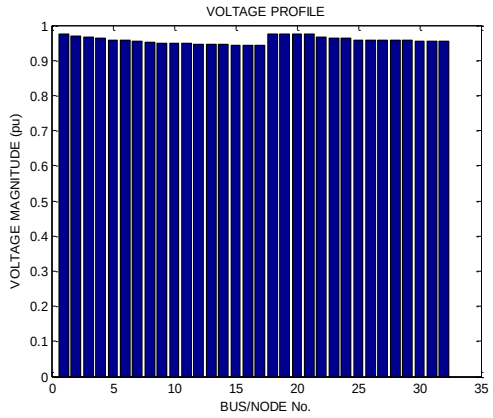


(a)

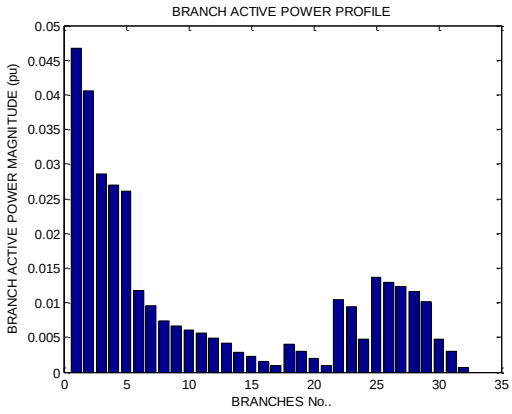


(b)

Figure 4. (a) Voltage Profile for 30-Nodes Network; (b) Branch Power Profile for 30-Node Network

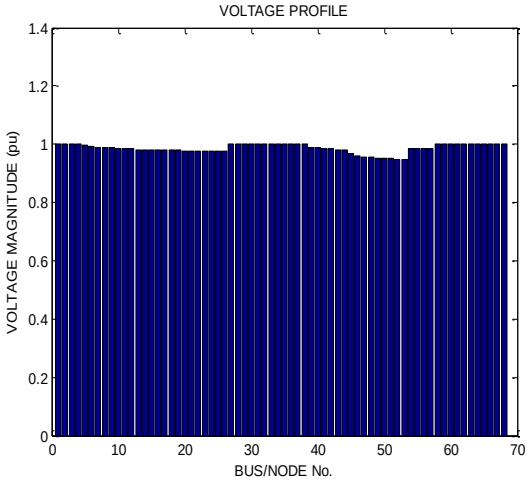


(a)

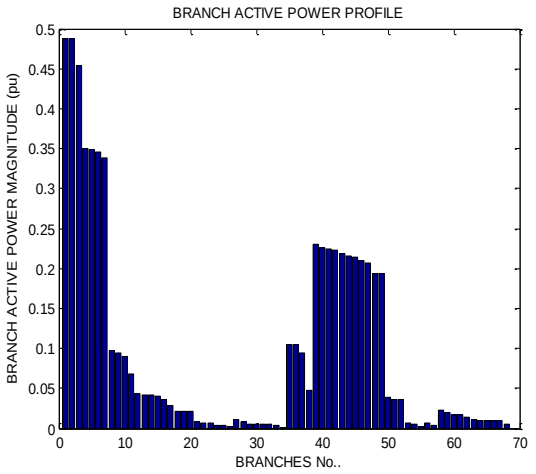


(b)

Figure 5. (a) Voltage Profile for 33-Nodes Network; (b) Branch Power Profile for 33-Node Network



(a)



(b)

Figure 6. (a) Voltage Profile for 69-Nodes Network; (b) Branch Power Profile for 69-Node Network

TABLE II. THE COMPLEX BUS VOLTAGES FOR THE 30, 33, AND 69 NODES/BUSES IEEE STANDARD NETWORKS

Bus Number	30-Bus	33-Bus	69-Bus
1	1	1	1
2.0000	0.9930 – 0.0092i	0.9759 -0.0162i	1.0000
3.0000	0.9869 - 0.0171i	0.9869 - 0.0171i	0.9869 - 0.0171i
4.0000	0.9769 - 0.0272i	0.9648 - 0.0351i	1.0001 - 0.0002i
5.0000	0.9682 - 0.0358i	0.9612 - 0.0416i	1.0000 - 0.0010i
6.0000	0.9599 - 0.0438i	0.9564 - 0.0593i	0.9957 - 0.0089i
7.0000	0.9537 - 0.0488i	0.9569 - 0.0643i	0.9912 - 0.0170i
8.0000	0.9504 - 0.0513i	0.9533 - 0.0673i	0.9901 - 0.0189i
9.0000	0.9466 - 0.0543i	0.9500 - 0.0725i	0.9899 - 0.0192i
10.0000	0.9447 - 0.0558i	0.9470 - 0.0772i	0.9868 - 0.0232i
11.0000	0.9440 - 0.0563i	0.9464 - 0.0777i	0.9861 - 0.0241i
12.0000	0.9438 - 0.0565i	0.9453 - 0.0787i	0.9841 - 0.0266i
13.0000	0.9865 - 0.0175i	0.9422 - 0.0838i	0.9821 - 0.0289i
14.0000	0.9861 - 0.0178i	0.9414 - 0.0861i	0.9803 - 0.0311i
15.0000	0.9860 - 0.0179i	0.9406 - 0.0873i	0.9784 - 0.0333i
16.0000	0.9860 - 0.0179i	0.9397 - 0.0884i	0.9788 - 0.0350i
17.0000	0.9533 - 0.0502i	0.9390 - 0.0905i	0.9783 - 0.0357i
18.0000	0.9477 - 0.0554i	0.9386 - 0.0910i	0.9782 - 0.0357i
19.0000	0.9412 - 0.0609i	0.9757 - 0.0168i	0.9780 - 0.0361i
20.0000	0.9360 - 0.0654i	0.9741 - 0.0203i	0.9778 - 0.0363i
21.0000	0.9314 - 0.0692i	0.9738 - 0.0211i	0.9775 - 0.0367i
22.0000	0.9258 - 0.0734i	0.9736 - 0.0218i	0.9775 - 0.0367i
23.0000	0.9212 - 0.0768i	0.9665 - 0.0316i	0.9774 - 0.0368i
24.0000	0.9169 - 0.0799i	0.9632 - 0.0378i	0.9773 - 0.0369i
25.0000	0.9148 - 0.0815i	0.9616 - 0.0408i	0.9772 - 0.0370i
26.0000	0.9140 - 0.0821i	0.9556 - 0.0612i	0.9772 - 0.0371i
27.0000	0.9138 - 0.0822i	0.9547 - 0.0638i	0.9772 - 0.0371i
28.0000	0.9531 - 0.0493i	0.9539 - 0.0750i	1.0001 - 0.0001i
29.0000	0.9527 - 0.0496i	0.9534 - 0.0831i	1.0001 - 0.0002i
30.0000	0.9524 - 0.0498i	0.9524 - 0.0868i	1.0000 - 0.0003i
31.0000	0	0.9506 - 0.0906i	1.0000 - 0.0003i
32.0000	0	0.9503 - 0.0915i	0.9999 - 0.0004i
33.0000	0	0.9503 - 0.0918i	0.9998 - 0.0006i
34.0000	0	0	0.9996 - 0.0009i
35.0000	0	0	0.9995 - 0.0009i
36.0000	0	0	1.0001 - 0.0002i
37.0000	0	0	1.0004 - 0.0017i
38.0000	0	0	1.0015 - 0.0061i
39.0000	0	0	1.0016 - 0.0067i
40.0000	0	0	1.0016 - 0.0067i
41.0000	0	0	0.9863 - 0.0241i
42.0000	0	0	0.9849 - 0.0265i
43.0000	0	0	0.9833 - 0.0293i
44.0000	0	0	0.9811 - 0.0331i
45.0000	0	0	0.9789 - 0.0368i
46.0000	0	0	0.9646 - 0.0546i
47.0000	0	0	0.9574 - 0.0633i
48.0000	0	0	0.9546 - 0.0666i
49.0000	0	0	0.9512 - 0.0704i
50.0000	0	0	0.9475 - 0.0765i
51.0000	0	0	0.9473 - 0.0767i
52.0000	0	0	0.9471 - 0.0770i
53.0000	0	0	0.9461 - 0.0786i
54.0000	0	0	0.9458 - 0.0791i
55.0000	0	0	0.9860 - 0.0241i
56.0000	0	0	0.9860 - 0.0241i
57.0000	0	0	0.9839 - 0.0269i
58.0000	0	0	0.9839 - 0.0269i
59.0000	0	0	1.0001 - 0.0001i

Bus Number	30-Bus	33-Bus	69-Bus
60.0000	0	0	1.0001 - 0.0003i
61.0000	0	0	1.0001 - 0.0005i
62.0000	0	0	1.0001 - 0.0005i
63.0000	0	0	1.0001 - 0.0005i
64.0000	0	0	1.0000 - 0.0012i
65.0000	0	0	1.0000 - 0.0015i
66.0000	0	0	1.0000 - 0.0016i
67.0000	0	0	1.0000 - 0.0016i
68.0000	0	0	1.0000 - 0.0017i
69.0000	0	0	1.0000 - 0.0017i

TABLE III. THE SIMULATION RESULTS FOR THE 30, 33, AND 69 NODES/BUSES IEEE STANDARD NETWORKS

Parameter	30-Bus	33-Bus	69-Bus
Simulation Time (CPU sec.)	1.0572	0.97568	4.7507
Number of Iterations	4	4	4
Total Real Power Loss (kW)	782	206	2098
Total Reactive Power Loss (kVAr)	238	103	952

V. CONCLUSION

An efficient power flow model has been proposed. A Matlab GUI based power flow simulator have been designed and programmed based on the proposed power flow model. The effectiveness developed simulator has been demonstrated using three IEEE standard test networks (30, 33, and 69 buses). The developed simulator looks promising in address issues arising from power flow computations.

REFERENCES

[1] Ashokumar, R. and P. Aravindhababu, *An improved power flow technique for distribution systems.* J Comput Sci, Informa Electr Eng, 2009. **3**(1): p. 1-8.

[2] Tinney, W.F. and C.E. Hart, *Power flow solution by Newton's method.* Power Apparatus and Systems, IEEE Transactions on, 1967(11): p. 1449-1460.

[3] Stott, B. and O. Alsaç, *Fast decoupled load flow.* power apparatus and systems, ieee transactions on, 1974(3): p. 859-869.

[4] Chen, T.-H., et al., *Distribution system power flow analysis-a rigid approach.* Power Delivery, IEEE Transactions on, 1991. **6**(3): p. 1146-1152.

[5] Zhang, F. and C.S. Cheng, *A modified Newton method for radial distribution system power flow analysis.* Power Systems, IEEE Transactions on, 1997. 12(1): p. 389-397.

[6] Aravindhababu, P. and R. Ashokkumar, *A fast decoupled power flow for distribution systems.* Electric Power Components and Systems, 2008. 36(9): p. 932-940.

[7] Eminoglu, U. and M.H. Hocaoglu, *A new power flow method for radial distribution systems including voltage dependent load models.* Electric power systems research, 2005. 76(1): p. 106-114.

[8] Wu, W. and B. Zhang, *A three-phase power flow algorithm for distribution system power flow based on loop-analysis method.* International Journal of Electrical Power & Energy Systems, 2008. 30(1): p. 8-15.

[9] Lin, W.-M. and J.-H. Teng, *Three-phase distribution network fast-decoupled power flow solutions.* International Journal of Electrical Power & Energy Systems, 2000. 22(5): p. 375-380.

[10] Shirmohammadi, D., et al., *A compensation-based power flow method for weakly meshed distribution and transmission networks.* Power Systems, IEEE Transactions on, 1988. **3**(2): p. 753-762.

[11] Aravindhababu, P., S. Ganapathy, and K. Nayar, *A novel technique for the analysis of radial distribution systems.* International journal of electrical power & energy systems, 2001. 23(3): p. 167-171.

[12] Milano, F., *An open source power system analysis toolbox.* Power Systems, IEEE Transactions on, 2005. 20(3): p. 1199-1206.

[13] Schoder, K., A. Hasanovic, and A. Feliachi, *PAT: a power analysis toolbox for MATLAB/Simulink.* Power Systems, IEEE Transactions on, 2003. 18(1): p. 42-47.