The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 11||Pages|| 05-12||2013||
ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805

On the Rate of Convergence of Newton-Raphson Method

Ranbir Soram[1], Sudipta Roy[2], Soram Rakesh Singh[3], Memeta Khomdram[4], Swavana Yaikhom[4], Sonamani Takhellambam[1]

[1]*Manipur Institute of Technology, Takyelpat, Imphal-795004*
[2]*Department of Information Technology, Assam University, Silchar-788011*
[3]*Rajiv Gandhi Institute of Technology, Bangalore-560032*
[4]*National Institute of Electronics and Information Technology, Akampat, Imphal-795001*

ABSTRACT

A computer program in Java has been coded to evaluate the cube roots of numbers from 1 to 25 using Newton-Raphson method in the interval [1, 3]. The relative rate of convergence has been found out in each calculation. The lowest rate of convergence has been observed in the evaluation of cube root of 16 and highest in the evaluation of cube root of 3. The average rate of convergence of Newton-Raphson method has been found to be 0.217920.

KEYWORDS : *Computer Program, Cube Root, Java, Newton-Raphson Method, Rate of Convergence.*

Date of Submission: 21, October, 2013 ⟺ Date of Acceptance: 10, November, 2013

I. INTRODUCTION

Numerical analysis is a very important branch of computer science that deals with the study of algorithms that use numerical approximation in mathematical analysis. It involves the study of methods of computing numerical data. It has its applications in all fields of engineering and the physical sciences including the life sciences. The stochastic differential equations and Markov chains have been used in simulating living cells for medicine and biology. Numerical analysis computes numerical data for solving numerically the problems of continuous mathematics. So, it implies producing a sequence of approximate values; thus the questions involve the rate of convergence, the accuracy of the answer, and even time consumed to report the answer. Many problems in mathematics can be reduced in polynomial time to linear algebra, this too can be studied numerically. Root-finding algorithms are also used to solve nonlinear equations. If the function is differentiable and the derivative is known and not equal to zero, then Newton-Raphson method is a popular choice. Linearization is another technique for solving nonlinear equations. The formal academic area of numerical analysis varies from quite foundational mathematical studies to the computer science issues involved in the creation and implementation of several algorithms.

NEWTON-RAPHSON METHOD

In numerical analysis, Newton-Raphson method is a very popular numerical method used for finding successively better approximations to the zeroes of a real-valued function $x : f(x) = 0$. Even though the method can also be extended to complex functions, we shall restrict ourselves to real-valued functions only. Newton-Raphson method has been used by a large class of users as it works very well for a large variety of equations like polynomial, rational, transcendental, trigonometric, and so on. It is also well suited on computers as it is iterative in nature. This feature of Newton-Raphson method has attracted many scientists and many scientific application programs use Newton-Raphson method as one of the root finding tools.The Newton-Raphson method in one variable is implemented as follows: Given a function $f(x)$ defined over the real x, and its derivative $f'(x)$, we begin with a first guess x_0 for a root of the function f. Provided the function satisfies all the assumptions made in the derivation of the formula, a better approximation x_1 is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \qquad (1)$$

Geometrically, $(x_1, 0)$ is the intersection with the x-axis of a line tangent to f at $(x_0, f(x_0))$.
The process is repeated as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (2)$$

until a sufficiently desired accurate value is obtained.

The idea of the method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the *x*-intercept of this tangent line (which is easily done with elementary algebra). This *x*-intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated. Suppose $f:[a,b] \to \mathrm{R}$ is a differentiable function defined on the interval [*a*, *b*] with values in the real numbers **R**. The formula for converging on the root can be easily derived. Suppose we have some current approximation x_n. Then we can derive the formula for a better approximation, x_{n+1} by referring to the fig. 1 given below. We know from the definition of the derivative at a given point that it is the slope of a tangent at that point.

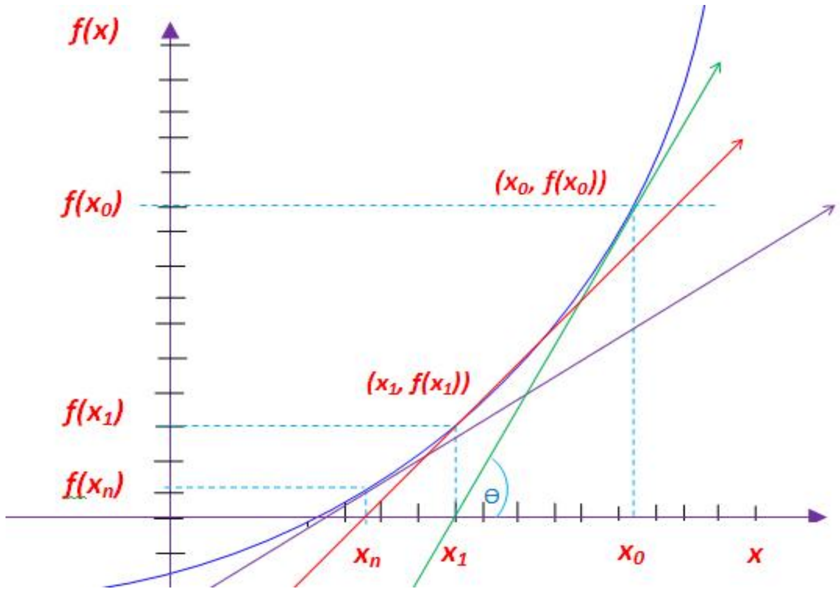

Fig. 1: Geometrical Interpretation of Newton-Raphson method

That is

$$f'(x_n) = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{x_n - x_{n+1}} \qquad (3)$$

Here, f' denotes the derivative of the function *f*. Then by simple algebra we can derive

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (4)$$

We start the iteration with some arbitrary initial value x_0. The method will usually converge, provided this initial guess is close enough to the unknown root, and that $f'(x) \neq 0$. We reproduce here the algorithm for Newton-Raphson method and it is from [2].

Algorithm 1. Newton-Raphson Method

1 Read x_0, epsilon, delta, n

Remarks: Here x_0 is the initial guess, epsilon is the prescribed relative error, delta is the prescribed lower bound for f' and *n* is the maximum number of iterations to be allowed. Statements 3 to 8 are repeated until the procedure converges to a root or iterations equal n.

2 *for* $i = 1$ to n in steps of 1 *do*

3 $f_0 \leftarrow f(x_0)$

4 $f_0^{'} \leftarrow f_0^{'}(x_0)$

5 *if* $\left| f_0^{'} \right| \leq$ delta *then* goto 11

6 $x_1 \leftarrow x_0 - \left(\frac{f_0}{f_0^{'}} \right)$

7 *if* $\left| \frac{x_1 - x_0}{x_1} \right| \leq$ epsilon *then* goto 13

8 $x_0 \leftarrow x_1$

endfor

9 Write " does not converse in ***n*** iterations", $f_0, f_0^{'}, x_0, x_1$

10 Stop

11 Write " slope too small ", $x_0, f_0, f_0^{'}, i$

12 Stop

13 Write " convergent solution", $x_1, f(x_1), i$

14 stop

II. ESTIMATING THE RATE OF CONVERGENCE

Let us take a sequence $x_1, x_2, ..., x_n$. If the sequence converges to a value r and if there exist real numbers λ > 0 and α ≥ 1 such that $\lim_{n \to \infty} \frac{|x_{n+1} - r|}{|x_n - r|^{\alpha}} = \lambda$ (5)

then we say that α is the rate of convergence of the sequence. Many root-finding algorithms are in fact fixed-point iterations. These iterations have the name fixed-point because the desired root r is a fixed point of a function g(x), i.e., $g(r) \to r$. For fixed-point iterations, if a value close to r is substituted into g(x) then the result is even closer to r so we are tempted to write $x_{n+1} = g(x_n)$.

It might be very convenient to define the error after n steps of an iterative root-finding algorithm as $e_n = x_n - \mathrm{r}$. As n → ∞ we see from equation (5) that

$$|e_{n+1}| \approx \lambda |e_n|^{\alpha} \quad (6)$$

$$|e_n| \approx \lambda |e_{n-1}|^{\alpha} \quad (7)$$

Dividing equation (6) by equation (7) we get the following formula given in equation (8).

$$\frac{|e_{n+1}|}{|e_n|} \approx \frac{\lambda |e_n|^{\alpha}}{\lambda |e_{n-1}|^{\alpha}} \approx \left| \frac{e_n}{e_{n-1}} \right|^{\alpha} \quad (8)$$

Now we solve for α and it gives

$$\alpha \approx \frac{\log \left| \frac{e_{n+1}}{e_n} \right|}{\log \left| \frac{e_n}{e_{n-1}} \right|} \quad (9)$$

Using the formula given in equation (9), we can determine the convergence rate α from two consecutive error ratios. But there is still a difficulty with this approach as to compute e_n we would need to know the root r which in fact we don't know. Anyway we can use the ideas above to develop a formula to estimate the rate of convergence of any method that behaves like a fixed-point method. If the method converges, then the sequence x_n it generates must satisfy the relation $|x_{n+1} - r| < |x_n - r|$, for n > N where N is some positive integer. As mentioned above, to compute $e_n = x_n - r$ we need to know the value of the root r which we still do not know. We will overcome this by assuming that the ratio of consecutive errors can be approximated by the ratio of consecutive differences of the estimates of the root:

$$\frac{e_{n+1}}{e_n} = \frac{x_{n+1} - r}{x_n - r} \approx \frac{x_{n+1} - x_n}{x_n - x_{n-1}} \qquad (10)$$

This allows us to approximate α with

$$\alpha \approx \frac{\log\left|\dfrac{x_{n+1} - x_n}{x_n - x_{n-1}}\right|}{\log\left|\dfrac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}\right|} \qquad (11)$$

Even though this only gives an estimate of α, in practice it agrees well with the theoretical convergence rates of Newton's method and gives us a good measure of the efficiency of various forms of fixed point algorithms.

Another formula that we can approximate the convergence rate α of Newton-Raphson method is

$$\alpha = \text{scaling factor X} \frac{1}{\text{No. of Iterations X } |f(r)|} \qquad (12)$$

where r is the estimated root and in case $f(r) = 0$, the formula fails.

III. WHY DO WE USE NEWTON-RAPHSON METHOD

The beauty of Newton-Raphson method lies in its efficiency and accuracy. The number of correct digits in each intermediate solution roughly at least doubles in every iteration and thus, a very high accuracy is obtained very quickly and it can be judged from the fig. 2. Thus, the use of Newton-Raphson method is highly demanding. In a problem where the Newton-Raphson method converges in one iteration, it amounts to using and evaluating one single formula: $x_{root} = x_0 - \frac{f(x_0)}{f'(x_0)}$ in order to obtain the desired root. Rather than solving any equations the users are just evaluating functions. In practice a single evaluation of the formula may fail to give the root, but two successive evaluations may give the solution. This again amounts to using the formula twice and is still reasonable and attractive. Although the functions involved may not converge so easily or the values of their derivatives at some points are too small to be used in the formula, a sufficiently small neighborhood can be found out in which they are almost convergent and the values of their derivatives at that neighborhoods can fit in the formula. By focusing our attention to such a neighborhood, Newton-Raphson method would yield the desired root in two iterations, if not in one. The starting point plays a very important role in the working of Newton-Raphson method. In general, a starting value far away from the actual solution may take more iteration to get the desired root. Sometime it may also happen that the iterations oscillate around the solution refusing to converge to it. There are many ways to avoid these conditions. Generally, choosing the initial guess sufficiently close to the solution is attractive. The closer to the zero of the function, the better it is. But, in the absence of any information about where the root might lie, a guess method might narrow the possibilities to a reasonably small interval by using the intermediate value theorem.

Table 1: Rate of convergence of Newton-Raphson method in the calculation of roots of $f(x)=x^3-n=0$.

S.No	Function	No. of Iteration	Value of the Root	Rate of convergence	Time taken in ms
1	$f(x)=x^3-1=0$	**1**	**1.000000000000000000000000000000000000000000000000000000000000000000000**	**5.000000**	**0**
2	$f(x)=x^3-2=0$	7	1.259921049894873164767210607278228350570251464701507980081975112155300	1.058113	14
3	$f(x)=x^3-3=0$	8	1.442249570307408382321638310	2.089114	16

			7801095883918692534993505775464161945 41688		
4	***f(x)=x³-4=0***	**5000**	**1.587401051968199474751705639272308260391493327899853009808285761825216**	**0.003202**	**1310**
5	*f(x)=x³-5=0*	9	1.7099759466766969893531088725438601098680551105430549243828617 07444296	0.210722	31
6	*f(x)=x³-6=0*	9	1.8171205928321396588912117563272605024282104631412196714813342 97931310	0.185394	16
7	*f(x)=x³-7=0*	9	1.9129311827723891011991168395487602828624390503458757662106476 40447234	0.167897	16
8	***f(x)=x³-8=0***	**9**	**2.000000000000000000000000000000000000000000000000000000000000000000000**	**5.000000**	**15**
9	*f(x)=x³-9=0*	9	2.0800838230519041145300568243578853863378053403732621096975910 80200106	0.100250	16
10	*f(x)=x³-10=0*	9	2.1544346900318837217592935665193504952593449421921085824892355 06346411	0.158022	15
11	*f(x)=x³-11=0*	10	2.2239800905693155211653633767221571965186991280969230556993458 08660401	0.117827	15
12	***f(x)=x³-12=0***	**5000**	**2.289428485106663735616084423879354017831813841575862144198104348131349**	**0.000235**	**1389**
13	***f(x)=x³-13=0***	**5000**	**2.351334687720757489500016339956914526915841983462175105040254311588343**	**0.000235**	**1357**
14	*f(x)=x³-14=0*	10	2.4101422641752299861283696676032728953545812899808676541641397 10413292	0.078721	15
15	*f(x)=x³-15=0*	10	2.4662120743304701014916113231545890427354844866280537601787874 10293377	0.061548	16
16	***f(x)=x³-16=0***	**5000**	**2.519842099789746329534421214556456701140502929403015960163950224310599**	**0.000185**	**1357**
17	*f(x)=x³-17=0*	10	2.5712815906582353554531872087397261164279016324546962598480223 76219940	0.193048	31
18	*f(x)=x³-18=0*	10	2.6207413942088966071416612804419962702394276457236317251027738 05728700	0.050412	31
19	*f(x)=x³-19=0*	10	2.6684016487219448673396273719708303350958785691831018656642135 86945794	0.154323	32
20	*f(x)=x³-20=0*	10	2.7144176165949065715180894696794892048051077694890969572843654 42803308	0.050715	16
21	*f(x)=x³-21=0*	11	2.7589241763811206694657911083585215822527120860389360328065925 02162799	0.032744	31
22	*f(x)=x³-22=0*	11	2.8020393306553871206656773856658940175857982187692683169 7	0.041051	16

			9783733783750		
23	*f(x)=x^3-23=0*	11	2.843866979851565477695439400958651852764165172737048104653425230235441	0.072259	32
24	*f(x)=x^3-24=0*	11	2.884499140614816764643276621560219176783738506998701155092832389083375	0.030560	31
25	*f(x)=x^3-25=0*	11	2.924017738212866065506787360137922778530498635101030041422573510072560	0.155596	31
		Average rate of convergence		**0.217920**	

IV. RESULT AND DISCUSSION

We have calculated the cube roots of integers from 1 to 25 using Newton-Raphson method in the interval [1, 3] with an initial guess of **1** in each case. We have used equation (12) with a scaling factor of $1x10^{-8}$. "0.000000000000000000000000000000000000000000000000000000000000000000001" is the tolerance limit used in our program. The rate of convergence, value of roots of the functions, and time takens in millisecond to obtain the roots are shown in Table-2. In the calculation of cube root of 1 and 8, we have taken α as 5.0 as $f(r)$ equals 0 in each case. As this will affect the average rate of convergence, we have excluded them in the calculation of the average rate of convergence.

As an example, we choose to calculate a root of the equation $f(x) = x^3 - 21 = 0$. The graph showing the value of each intermediate value at each iteration is given in fig. 2. The graph showing the time taken in millisecond to calculate a zero of $f(x) = x^3 - n = 0$ for $n = 1$ to 25 is shown in fig. 3.The speed at which the program reports a root of the function is highly attractive. In most cases, the program has taken less than 33 milliseconds to calculate a root of the function. It is highly surprising as we use "0.000000000000000000000000000000000000000000000000000000000000000000001" as the tolerance limit in the program. We can improve the program a lot if we choose different initial guess for each function. The program executed very quickly in the calculation of cube root of 2 and it had taken 14 milliseconds to report a root. The most time consuming execution of the program was reported in the calculation of a cube root of 12 as it took 1389 milliseconds to report the root. The graph in fig. 4 shows the relative rate of convergence in the calculation of cube roots of integers from 1 to 25 of Newton-Raphson method in the interval [1, 3].

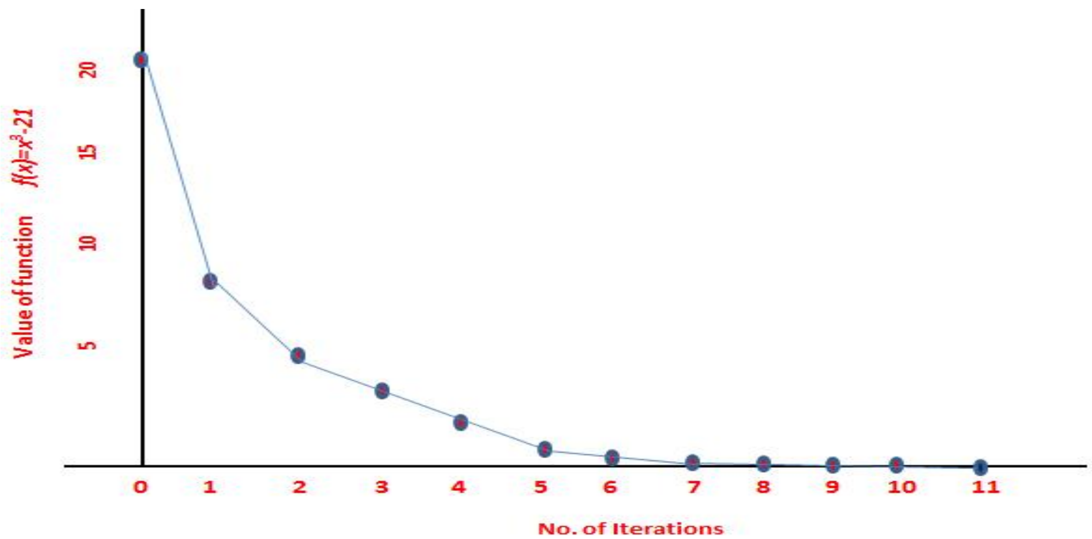

Fig. 2: Value of function f(x)=x^3-21 from iteration 1 to 11

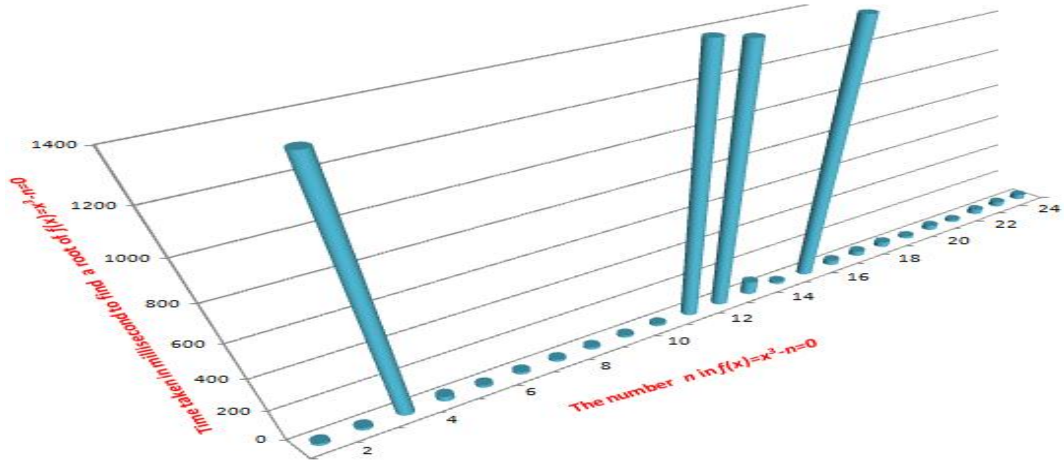

Fig. 3: Time taken in millisecond to calculate a root of $f(x)=x^3-n=0$

V. CONCLUSION

The highest rate of convergence of Newton-Raphson method has been observed in the calculation of cube root of 3 and is equal to 2.089114. The lowest rate of convergence of Newton-Raphson method has been observed in the calculation of cube root of 16 and is equal to 0.000185. Average rate of convergence of Newton-Raphson method is calculated to be 0.217920.

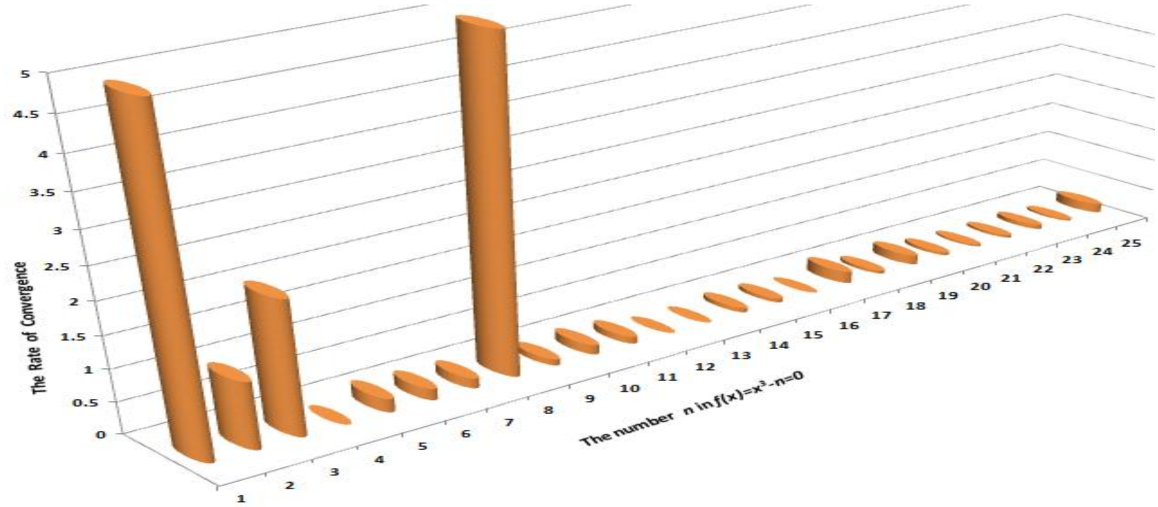

Fig. 4: Rate of convergence in reporting a root of $f(x)=x^3-n=0$

ACKNOWLEDGEMENTS

The first author thanks his daughter Java and son Chandrayan for not using his laptop during the preparation of the manuscript. The fourth author thanks her husband for his understanding and support in the preparation of the manuscript. The fifth author thanks her family and teachers for their supports in the writing of this manuscript.

REFERENCES

[1] Zoltan Kovacs, *Understanding convergence and stability of the Newton-Raphson method* (Department of Analysis, University of Szeged, 2011).
[2] V. Rajaraman, C*omputer Oriented Numerical Method (*Prentice-Hall of India, 2005).
[3] E Balagurusamy, *Computer Oriented Statistical and Numerical Methods*(Macmillan India Limited, 1988)
[4] C. Woodford , C. Phillips, *Numerical Methods with Worked Examples: Matlab Edition* (Springer London , 1997).
[5] N. Daili, *Numerical Approach to Fixed Point Theorems*, Int. J. Contemp. Math. Sciences, Vol. 3, 2008, no. 14, 675 – 682.

Ranbir Soram works at Manipur Institute of Technology,Takyelpat, Imphal, India. His field of interest includes Network Security, NLP, Neural Network, Genetic Algorithm, and Fuzzy Logic etc.

Sudipta Roy works in the Department of Information Technology, Assam University Silchar.

Soram Rakesh Singh is a PG student in the Department of Computer Science & Engineering at Rajiv Gandhi Institute of Technology, Bangalore.

Memeta Khomdram works at National Institute of Electronics and Information Technology (Formerly DOEACC Centre), Akampat, Imphal.

Swavana Yaikhom **w**orks at National Institute of Electronics and Information Technology, Akampat, Imphal.

Sonamani Takhellambam works at Manipur Institute of Technology,Takyelpat, Imphal, India.