Automated Proving Properties of Expectation-Maximization Algorithm Using Symbolic Tools

Vladimir Mladenovic, *Member*, *IEEE*, Maja Lutovac, and Miroslav Lutovac, *Senior Member*, *IEEE*

***Abstract* — In many analyses based on estimating the parameters of probability distribution functions, the algorithms are developed for unknown probabilities. Some algorithms are derived starting from previous solutions and algorithms. One very popular algorithm is the EM (Expectation-Maximization) algorithm. The EM algorithm is a starting point for developing other advanced algorithms. Features of EM and other algorithms are observed with the traditional numerical approach. In this paper, we present a new approach of analysing the EM algorithm using computer algebra tools (Mathematica). We automatically derive properties of the algorithm. The knowledge embedded in symbolic expressions was used to simulate an example system and EM algorithm to generate the implementation code and to understand the nature of the error produced by selecting the total number of observed elements.**

***Keywords* — Convergence, EM algorithm, iteration, ML estimation, symbolic processing.**

I. Introduction

COMPUTER algebra is a field of computer science concerned with the development, implementation, and application of algorithms that manipulate and analyse expressions. For the past 40 years, research has been concerned with the development of effective and efficient algorithms for many mathematical operations [1]. Some of the issues are developing algorithms (such as algebraic algorithms, symbolic-numeric algorithms), studying how to build computer algebra systems (memory management, higher-order type systems, optimizing compilers), and mathematical knowledge management (representation of mathematical objects). Computer algebra systems can be used to simplify rational functions, factor polynomials, find the solutions to a system of equations, give general formulas as answers, model industrial mathematical problems, and various other manipulations. Computer algebra deals with arithmetic on defined algebraic structures, polynomials, matrices, algebraic functions, and may involve symbols parameters and indeterminates. Symbolic computation can be treated as a transformation of expression trees, such as symbols for operations ("+", "sin"), variables, constants, and simplification (for example, expression equivalence). Computer algebra is based on well-defined semantics, compose constructions, and algebraic algorithms. Symbolic computation is concerned with alternative forms (factored, expanded), partially-specified domains, and symbolical derivations in general.

The purpose of this paper is to extend the application of computer algebra in the field based on statistics theory, particularly for discovering and proving properties.

The EM (Expectation-Maximization) algorithm is analysed with a new type of processing, called symbolic processing. We illustrate the main drawbacks of numerical tools that have led to erroneous conclusions in the application of the algorithm through a practical example. Using symbolic processing, we derive the error function in closed form as a function of the number of iterations. Next, we determine the minimum number of iterations required to obtain a correct result, and as well as the required number of iterations as a function of the number of accurate bits of the final result of an algorithm. This approach should enable better understanding of the main features of the algorithm, prove its convergence in time domain and reduce the number of iterations. The EM algorithm is presented at a level suitable for signal processing practitioners who have had some exposure to estimation theory. A starting point of EM algorithm is the ML (Maximum Likelihood) estimation, and many practical numerical problems presented in [2]. Two basic limitations of EM algorithm are (1) slow convergence and (2) maximization steps difficulties [3]. Usually, researchers have mainly focused on solving these problems from the aspect of a traditional numerical approach. Also, some new algorithms based on the EM algorithm can benefit from this new approach using the symbolic processing.

II. Symbolic Processing

Modern research in the field of engineering science often includes two phases - measurements and simulations. With comparable results from both phases relevant conclusions can be drawn for the implementation and/or further investigation. Very often, the large number of input parameters that are set numerically in the simulation process leads to disagreement in the results [4]. Some of the reasons for these disagreements can be a finite data word-length and the imperfections of certain numerical tools and algorithms [5]. The experience often shows an

This work is supported in part by the Ministry of Education and Science of Serbia under Grant TR32023.

Vladimir Mladenovic is with the Higher Technical School of Professional Studies, Nemanjina 2, 12000 Požarevac, Serbia (phone: 381-64-1268752; e-mail: vlada@open.telekom.rs).

Maja Lutovac is with the Lola Institute, Kneza Višeslava 70A, 11030 Belgrade, Serbia (e-mail: majalutovac@yahoo.com).

Miroslav Lutovac is with the University Singidunum, Danijelova 32, 11000 Belgrade, Serbia, (phone: 381-62-8132280; e-mail: mlutovac@singidunum.ac.rs).

inconsistency of measured and simulated results that can be very confusing for researchers.

One of the ways to overcome accuracy problems in simulations and processing is to use symbols instead of numbers, so that the influence of symbolic parameters can be easily recognized in the final result.

Symbolic processing may play an important role in education for better understanding of the nature of the analysed phenomena. Also, it can be used in practice in order to prevent the accuracy problems introduced by numeric tools.

This paper aims to highlight the role of symbolic processing applications, especially when the numeric approaches may produce incorrect conclusions. Therefore, symbolic processing is important in many areas - engineering and especially in telecommunications.

Symbolic processing is a unique feature that performs computations as closed-form expressions in terms of system parameters kept as symbols. Even more, for a symbolic input sequence, one can compute the symbolic output sequence with system parameters specified by symbols. All system parameters and the initial conditions can be given by symbols and the derived result is the most general, and all calculations are generated automatically. Numerical processing may show unsatisfactory results as in [6]. An inexperienced user can reach wrong conclusions from one of the most cited paper [6] as it will be presented in this paper. Using symbolic tools, it is easy to prove that many successive errors in [6], in the derivation of intermediate results, finally were completed with an example with accurate results. The average user cannot repeat the derivation process because it is not correct, although the final numeric result in paper [6] is correct.

Symbolic processing works with symbols, while numbers are only a special case of symbols that perform some operations such as addition and multiplication operations. Therefore, writing programs using symbolic processing can be seen as a set of instructions that manipulate the symbols and can be used to perform a much wider range of activities.

In the next section we briefly repeat the basic theory presented in [6]. Next, we import the knowledge into the computer algebra system. Finally, we automate the derivation of expressions from [6] and derive accurate formulas. Finally, the code used for processing is also automatically generated from the derived results.

III. Multinomial Distribution, Conditional Probability and EM Algorithm

We start with the multinomial distribution and conditional probability [7], [8]. It is important to emphasize the role of the EM algorithm because this is the backbone for future developed algorithms. For example it is used in analysing measured data and simulations [3], [9].

The EM algorithm consists of two steps: the E-step (Expectation step) and M-step (Maximization step). The E-step computes the expected value of a variable given according to a current estimate of a parameter and observed data. A general form of this step is presented in [6]:

$$Q(p|p^{[k]}) = E[\log f(x|p)|y, p^{[k]}], \tag{1}$$

where $f(x|p)$ is the probability density function of complete data, p is a set of parameters of the density and $y \in \Re^m$ is a many-to-one mapping. The Q-function is denoted as $Q(\cdot)$ and k is the number of iterations.

The M-step uses data from the E-step and it gets the values of unknown parameter of distribution p using ML estimation:

$$p^{[k+1]} = \arg\max_p Q(p|p^{[k]}). \tag{2}$$

Starting with an initial value, $p^{[0]}$, in the next iterations new $p^{[k+1]}$ is computed for known $p^{[k]}$, $k \in \{0,1,2,...\}$. After that the E-step and M-step are performed until convergence is obtained, and then the computation stops when $\left\|p^{[k]} - p^{[k-1]}\right\| < \varepsilon$ (where ε is a given appropriate accuracy).

IV. Analysis of the EM Algorithm

In this paper, we re-derive the expressions obtained in [6] in order to find errors in the intermediate results. We use them in the same way as in [3]. The application of symbolic processing of the EM algorithm is illustrated through the same example used in [6].

The example problem is to find the unknown value of p using the EM algorithm. In this example the initial value of p is defined as a symbol, and we know that its true value is 1/2 (as specified in the example in [6]). Now, we define the distribution functions. According to the analysed case [6], the notation `f[.]` is used to indicate the probability function:

```
f[x1_, x2_,x3_,p1_,p2_,p3_]:=
(x1+x2+x3)!p1^x1/x1! p2^x2/x2! p3^x3/x3!
```
(3)

The notation `f2[.]` is used for the probability function of $y_1 = x_1 + x_2$ and $y_2 = x_3$:

```
f2[y1_, y2_,p1_,p2_,n_]:=n! p1^y1/y1! p2^y2/y2!
```
(4)

Based on [6], the probabilities (`p1`, `p2`, and `p3`) are known (`p1`, `p2`, and `p3` in terms of parameter `p`) for three types of objects (`x1`, `x2`, and `x3`) and also the total number of observed objects `n=x1+x2+x3` is also known for a particular case `n0`:

```
p1=1/4; p2=1/4+p/4; p3=1-p1-p2; n0=100;
subs1={p1→p1, p2→p2, p3→p3, n→n0}
```
(5)

Substituting parameters from (5) in (3) and (4), we obtain the conditional probability distribution:

```
fX1andX1plusX2=f[x1,x2,x3,p1,p2,p3]
%/.subs1
```

$$\frac{4^{-x1}\left(\frac{1}{2}-\frac{p}{4}\right)^{x3}\left(\frac{1}{4}+\frac{p}{4}\right)^{x2}(x1+x2+x3)!}{x1!\,x2!\,x3!} \tag{6}$$

Next, we derive the conditional probability distribution:

```
fX1plusX2 = f2[x1+x2,x3,p1+p2,p3,(x1+x2+x3)]
%/.subs1
```

$$\frac{\left(\frac{1}{2}-\frac{p}{4}\right)^{x3}\left(\frac{1}{2}+\frac{p}{4}\right)^{x1+x2}(x1+x2+x3)!}{(x1+x2)!\,x3!} \tag{7}$$

Using the symbol for the total number of observed objects `n=x1+x2+x3`, results (6) can be simplified to:

```
f[x1,x2,x3,p1,p2,p3]/.subs1/.(x1+x2+x3)→n
```

$$\frac{4^{-x1}\left(\frac{1}{2}-\frac{p}{4}\right)^{x3}\left(\frac{1}{4}+\frac{p}{4}\right)^{x2} n!}{x1!\,x2!\,x3!} \tag{8}$$

The result (8) is in consistency with [6] :

$$P(X_1=x_1,X_2=x_2,X_3=x_3|p,)= \left(\frac{n!}{x_1!x_2!x_3!}\right)\cdot\left(\frac{1}{4}\right)^{x_1}\cdot\left(\frac{1}{4}+\frac{p}{4}\right)^{x_2}\cdot\left(\frac{1}{2}-\frac{p}{4}\right)^{x_3} \tag{9}$$

A. Derivation of Conditional Probability: E-step

According to [6, Box 2], we derive expressions of conditional probabilities and conditional expectations. First, for x_1, assuming that $y=x_1+x_2$, the conditional probability is:

```
cp = fX1andX1plusX2/fX1plusX2 /.x2→y-x1/.subs1
```

$$\frac{(1+p)^{-x1+y}(2+p)^{-y}\,y!}{x1!\,(-x1+y)!} \tag{10}$$

The conditional expectation for x_1 is:

```
ex1 = Sum[x1 cp /.subs1, {x1,0,y}]
```

$$\frac{y}{2+p} \tag{11}$$

The result (11) is different from that presented in [6, Box 2]. Similarly, for x_2, conditional expectation is:

```
ex2 = Sum[x2(fX1andX1plusX2/fX1plusX2 /.x1→y-x2), {x2,0,y}]
%/.subs1// Simplify
```

$$\frac{(1+p)\,y}{2+p} \tag{12}$$

The result (12) is the same as in [6, Box 2]. The error in [6, Box 2] is obvious because it should be `ex1+ex2=y`, as derived by (11) and (12), while in [6] it is `ex1+ex2≠y`. Thus, we prove that conditional expectation for x_1 is not correct in [6].

B. Derivation of the First Derivative of log(f): M-step

The M-step is performed in the same manner as described in [6]. It is performed in two steps in a symbolic form: (1) we derive a derivative function and (2) we look for the extreme value based on the obtained derivative:

```
d = D[Log[fX1andX1plusX2/.subs1],p]
```

$$\frac{(-2+p)\,x2+(1+p)\,x3}{(-2+p)(1+p)} \tag{13}$$

```
sol1 = Solve[d==0,p]//Flatten
```

$$\left\{p\to\frac{2\,x2-x3}{x2+x3}\right\} \tag{14}$$

V. Symbolic Approach to Derive Correct Property

After successive application of E-step and M-step, the required parameter p is computed as an extreme point:

```
x2k[y1_,pk_]:=x2e/.{y→y1,p→pk}
pk1 = p/.sol1/.x2→x2k[y1,pk]//Simplify
```

$$pk1 = \frac{-(2+pk)\,x3+2(1+pk)\,y1}{(2+pk)\,x3+(1+pk)\,y1} \tag{15}$$

This is not in consistency with the result in [6], and it is obvious that the corresponding intermediate result to (15) in [6] is not correct:

$$p^{[k+1]}=\frac{p^{[k]}(4y_1-2x_3)+2y_1-2x_3}{p^{[k]}(2y_1-2x_3)+y_1+2x_3}. \tag{16}$$

A. Closed Form Solution in Terms of the Iteration Step instead of the Convergence Analysis

The convergence is very important to determine whether an algorithm comes to a final value. In order to satisfy the convergence, solving of the unknown parameter, it should be expressed in a closed form. We set the recurrence equation for $p^{[k+1]}$, and known $p^{[0]}=0$, so that the iterative procedure is given in the following way:

```
eq1={p[k+1]==(pk1/.pk→p[k]),p[0]==0}
```

$$\left\{p[1+k]==\frac{2\,y1\,(1+p[k])-x3\,(2+p[k])}{y1\,(1+p[k])+x3\,(2+p[k])},\ p[0]==0\right\} \tag{17}$$

```
sol2=RSolve[eq1,p[k],k]//Flatten
```

$$\left\{p[k]\to-\left(2\left(-3^k\left(\frac{-x3-y1}{x3}\right)^k+\left(\frac{-x3-y1}{y1}\right)^k\right)(x3-y1)\right)\Big/ \left(-3^k\,x3\left(\frac{-x3-y1}{x3}\right)^k+2\,x3\left(\frac{-x3-y1}{y1}\right)^k- 3^k\left(\frac{-x3-y1}{x3}\right)^k y1-2\left(\frac{-x3-y1}{y1}\right)^k y1\right)\right\} \tag{18}$$

The number of objects `x1[k]` and `x2[k]` in the `k`-th iteration can be computed using the following statements, as well as the new value of parameter `p[k]` is computed using (14) and `x3[k]=n-x1[k]+x2[k]`:

```
x1k[y1, pk] /. pk→p[k]
```

$$\frac{y1}{2+p[k]}$$

```
x2k[y1, pk] /. pk→p[k]
```

$$\frac{y1\,(1+p[k])}{2+p[k]}$$

```
subs2 = {x1→Round[n0 p1], x2→Round[n0 p2],
  x3→n0-Round[n0 p1]-Round[n0 p2]} /. p→p0
pkn = p[k] /. sol2 /. y1→(x1+x2) /. subs2
```

$$-\frac{13\left(37^k-189^k\right)}{13\times37^k+25\times189^k} \tag{19}$$

In order to test convergence, we compute the limit case:

```
Limit[pkn/.sol1,k→Infinity]
%//N
```
(20)

Finally, the result of processing should be 1/2, but we compute 13/25=0.52. We know the exact value in this example should be 0.5, but the derived result is 0.52, and even more in [6] the value of y_1 is 63, not 100 as it is given in the theoretical part of [6].

We can use results (15) and (19) to automatically set a processing code and perform an iterated procedure. The results of iterations can be viewed in a matrix form.

TABLE 1: NUMERICAL VALUES OF ALGORITHM.

K	x_1	x_2	p
1.000000	26.47546	36.52454	0.3795620
2.000000	25.29816	37.70184	0.4903000
3.000000	25.05874	37.94126	0.5140929
4.000000	25.01151	37.98849	0.5188400
5.000000	25.00225	37.99775	0.5197728
6.000000	25.00044	37.99956	0.5199555
7.000000	25.00009	37.99991	0.5199913
8.000000	25.00002	37.99998	0.5199983
9.000000	25.00000	38.00000	0.5199997
10.00000	25.00000	38.00000	0.5199999
11.00000	25.00000	38.00000	0.5200000
12.00000	25.00000	38.00000	0.5200000

The results from Table 1 show that the limit value is obtained after 11 iterations. Unfortunately, there is no explanation why the limiting value is not 0.5.

B. *Accurate Bits as a Function of Iterations*

The objective of this section is to calculate the minimum number of bits **b** needed to represent the correct result for **p** as a function of the number of iterations **k**. This is possible because we already have a closed form solution of the parameter **p** in terms of the iteration step **k**, for any combination of objects of different kinds, see (18). As an example, we will use the same numeric values as in [6]. The total number of observed objects is **n=100**, and the total number of observed objects of two kinds is **y1=63**, as in [6].

First, we substitute the known rules and numeric values in the derived function (18), and thus obtain the value of **p** in terms of **k**.

```
pkn = p[k] /. sol2 /. x3 → (n - y1) /. {n → 100, y1 → 63}
```

$$\left(52\left(-\left(-\frac{300}{37}\right)^k+\left(-\frac{100}{63}\right)^k\right)\right)\Big/ \left(37\left(-\frac{25}{63}\right)^k 2^{1+2k}-7\left(-\frac{100}{37}\right)^k 3^{2+k}-(-300)^k 37^{1-k}-(-25)^k 2^{1+2k} 63^{1-k}\right) \tag{21}$$

Next, we define the error function as a difference between the exact value after an infinite number of iterations and the current value in any iteration:

```
error =
 Limit[pkn /. v → n0, k → Infinity] - (pkn /. v → n0)
```

$$\frac{494}{325+625\left(\frac{189}{37}\right)^k} \tag{22}$$

The number of accurate bits can be expressed in terms of the number of iterations in a closed form.

```
sol3 = Solve[error == 1/2^b, b]//Flatten
```

$$\left\{b \to -\frac{\text{Log}\left[\frac{494}{325+625\left(\frac{189}{37}\right)^k}\right]}{\text{Log}[2]}\right\} \tag{23}$$

The result is graphically presented in Fig. 1.

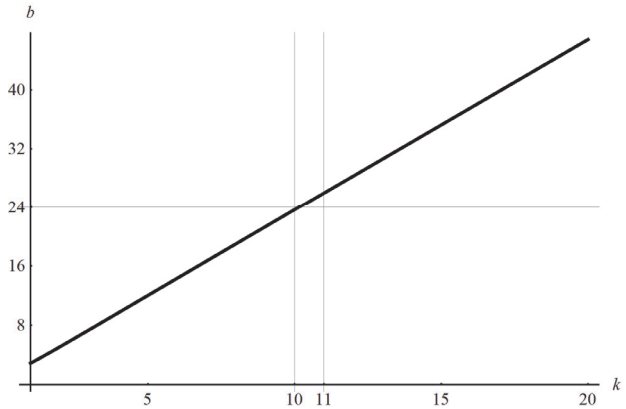

Fig. 1. Number of accurate bits b in terms of number of iterations k for 100 observed elements.

From Fig. 1 it seems that the number of accurate bits linearly increases. For a chosen number of iterations, it is easy to find the number of accurate bits. For example, after 10 iterations the number of accurate bits is 23, while 11 iterations will produce correct results with 24 bit accuracy.

C. *Number of Iterations as a Function of Accurate Bits*

In a similar way we can determine the minimum number of iterations **k** needed to represent the correct result of the parameter **p** as a function of the number of accurate bits **b**. The next expression shows the procedure to derive the closed form solution.

```
sol4 = Solve[error == 1/2^b, k]//Flatten
```

$$\left\{k \to \frac{\text{Log}\left[\frac{1}{4}\left(-2+3\times 2^b\right)\right]}{\text{Log}[5]}\right\} \tag{24}$$

Fig. 2 illustrates the number of accurate bits **b** as a function of the number of iterations, for 100 observed elements.

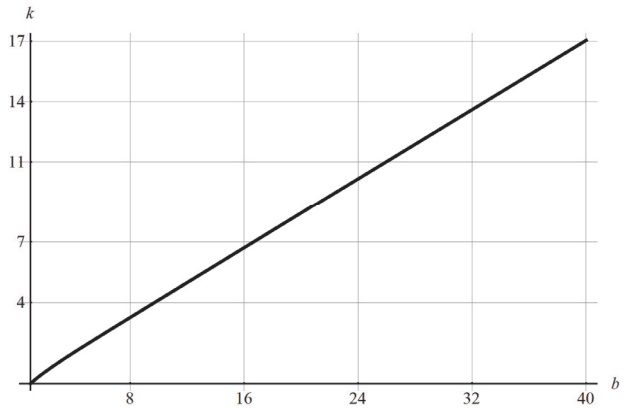

Fig. 2. Number of iterations k in terms of number accurate bits b for 100 observed elements.

For a specified number of accurate bits, it is easy to find the required number of iterations. For example, for 24 bit accurate result, 11 iterations are required.

D. *Iterations vs Accurate Digits*

In a similar way we can calculate the minimum number of iterations k needed to represent the correct result in terms of accurate digits. The procedure is as follows:

```
sol5 = Solve[error == 1/10^digits, k]
```

$$\left\{\left\{k \to \frac{\text{Log}\left[\frac{1}{4}\left(-2+3\times 10^{\text{digits}}\right)\right]}{\text{Log}[5]}\right\}\right\} \tag{25}$$

Solving the same equation, we can determine the number or accurate digits, **digits**, in terms of iteration step **k** for a known number of observed elements.

VI. Accuracy vs Number of Observed Elements

The main drawback in understanding the EM algorithm as explained in [6] is that the implementation code and the results obtained using that code are not in agreement with the properties of the statistic process. For example, the computed value of parameter **p** is 0.52, although we define the problem starting with **p=0.5**. Inexperienced users can conclude that this is due to the accuracy problem or that their implementation is somehow wrong.

The purpose of this section is to analyse the influence of the total number of observed objects and thus identify the difference of results obtained using the EM algorithm and the assumptions.

Let us repeat the same example for the same parameters presented in [6]. Assume that the total number of observed elements is **n0**, **pTrue** is the true value of **p**, and **y10** is the total number of elements of two different type objects. This can be described symbolically as follows:

```
pTrue = 1/2;
n0 = 100;

pkofNandY1 = p[k] /. sol2 /. x3 → (n - y1)

y10 = Round[1/4 n0] + Round[(1/4 + pTrue/4) n0];
pkn = Simplify[pkofNandY1 /. {n → n0, y1 → y10},
  Assumptions → Element[v, Integers]]
```

$$-\frac{13\left(\left(\frac{100}{63}\right)^k-\left(\frac{300}{37}\right)^k\right)}{13\left(\frac{100}{63}\right)^k+\left(\frac{12}{37}\right)^k 25^{1+k}} \tag{26}$$

```
pkn /. k → {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
N[%, 7]

{0.379562, 0.490300, 0.514093, 0.518840, 0.519773,
 0.519955, 0.519991, 0.519998, 0.520000, 0.520000,
 0.520000, 0.520000}
```

Although we have started with the parameter value 0.5, the algorithm produces the value 0.52. We can repeat the same code but with a different number of observed objects:

```
pTrue = 1/2;
n0 = 120;

y10 = Round[1/4 n0] + Round[(1/4 + pTrue/4) n0];
pkn = Simplify[pkofNandY1 /. {n → n0, y1 → y10},
  Assumptions → Element[v, Integers]]
```

$$\frac{-1+5^k}{1+2\times 5^k} \tag{27}$$

```
pkn /. k → {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
N[%, 7]

{0.363636, 0.470588, 0.494024, 0.498801, 0.499760,
 0.499952, 0.499990, 0.499998, 0.500000, 0.500000,
 0.500000, 0.500000}
```

A different number of observed objects will generate different final results for the same process. Therefore, we identify the reason for error of the algorithm as an error produced by quantizing the number of observed objects of one kind in the first case and in [6]. When the number of objects is correct with respect to the probability, then the final result is also accurate.

Notice that the code derived for implementation depends on the number of observed elements, and consequently the result will be different. This property is similar to leakage in digital signal processing. Fig. 3 presents the computed parameter **p** after 10 iterations for different numbers of observed elements.

The computed value of parameter using the EM algorithm oscillates around the correct value 0.5.

The error produced by the number of observed objects can be simply computed using the following code:

```
pknx = pkn /. {k → 10, n0 → Range[100, 120]}
{0.520, 0.495, 0.510, 0.524, 0.500, 0.476, 0.491,
 0.505, 0.481, 0.495, 0.509, 0.523, 0.500, 0.478,
 0.491, 0.504, 0.517, 0.496, 0.508, 0.521, 0.500}
```
(28)

The maximal value after 10 iterations for a number of observed objects larger than 100 is 0.524. The minimal value is 0.476.

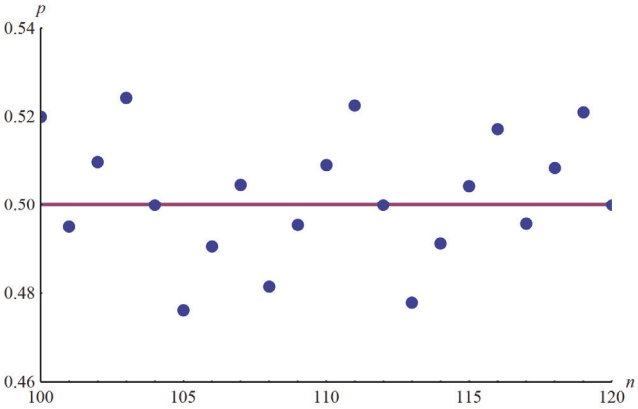

Fig. 3. Computed parameter p in terms of total number of observed elements n, after 10 iterations.

The above analysis shows that the error of the original algorithm after two iterations is lower than the error that

can be produced using an inadequate number of the total number of observed objects. Even more, with more iterations, we compute the final result with more accurate bits, or we can obtain more accurate results that are much more different than the correct value of the unknown parameter.

The understanding of the nature of the process under consideration can help in developing a more appropriate algorithm and we can avoid unnecessary computations.

VII. Conclusion

In this paper, we analysed the EM algorithm using a new approach named symbolic processing. In the EM algorithm, the traditional numerical approach has given approximated results and based on them some conclusions are derived. We prove that a more elegant and accurate solution can be obtained by using symbolic processing. Also, we provide better solutions for the exact values of the unknown parameters of probability.

On the basis of the conclusions and results, it is shown that the symbolic processing approach has several advantages over the numerical one.

Also, the symbolic approach gives a possibility for application in advanced algorithms that are developed using the EM algorithm, and it represents a new challenge for further research.

References

[1] J. S. Cohen, "Computer Algebra and Symbolic Computation – Mathematical Methods," Natick, Massachusetts, AK Peters. 2003.

[2] G. McLachlan and T. Krishnan, "The EM Algorithm and Extensions. Probability and Statistics," New York: Wiley, 1996.

[3] J. A. Fessler and A. O. Hero, "Space-alternating generalized expectation maximization algorithm", *IEEE Trans. Signal Processing*, vol. 42, pp. 2664–2677, Oct. 1994.

[4] V. Mladenovic, D. Porrat, and M. Lutovac, "The direct execution of the expectation-maximization algorithm using symbolic processing," *10th Int. Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, TELSIKS, Oct. 5-8, 2011. Niš, Serbia.

[5] M. D. Lutovac and D. V. Tošić, "Symbolic analysis and design of control systems using Mathematica," *International Journal of Control*, *Special Issue on Symbolic Computing in Control*, vol.79, no.11, pp.1368–1381, 2006.

[6] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Mag*, , pp. 47–60, 1997.

[7] Weisstein, Eric W. "Multinomial Distribution," MathWorld - A Wolfram Web Resource, Wolfram Research, Inc., 1999-2011. http://mathworld.wolfram.com/MultinomialDistribution.html.

[8] Weisstein, Eric W. "Conditional Probability," MathWorld - A Wolfram Web Resource, Wolfram Research, Inc., 1999-2011. http://mathworld.wolfram.com/ConditionalProbability.html.

[9] B. H. Fleury, D. Dahlhaus, R. Heddergott, and M. Tschudin, "Wideband angle of arrival estimation using the SAGE algorithm," in *Proc. IEEE Fourth Int. Symp*. *Spread Spectrum Techniques and Applications*, ISSSTA '96, Mainz, Germany, pp. 79–85, Sept. 1996.

[10] V. M. Mladenovic, D. Porrat, and M. D. Lutovac "Simulation of OFDM Transmitters and Post Processing with SchematicSolver and Mathematica as a Computer Algebra System," *5th European Conference on Circuits and Systems for Communications*, ECCSC 10, Belgrade, Serbia, November 23–25, 2010.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Stat. Soc. Series B*, vol. 39, no. 1, pp. 1-38, 1977.