General solution of problem (P vs. NP)

Grzegorz Ileczko

e-mail: igreg88@gmail.com

75-627 Koszalin, Poland

This article demonstrates a general solution for the problems of class (P vs.NP). Peculiarly for the problems of class (P=NP). Presented solution is quite simple and can be applicable in many various areas of science. At general, (P=NP) it's a class of problems which possess algorithmic nature. The algorithms should contains one or more of logical operations like (if...then) instruction, or Boolean operations. The proper proof for this thesis with a new formula was presented. Except formula, one proper example was presented for the problem (P=NP). Exists a lot of problems for which P class problems are equivalent with the NP problems (P=NP). Millions, I think.

For example, I discovered extremely effective algorithm for the *"Hamiltonian Path Problem"*. Algorithm can find the proper solution for 100 cities at very short time. Solution time for old laptop is less than two seconds. Classical solution for that problem exists, but is extremely difficult and computer's time is huge. Algorithm for the Hamilton problem, will be presented at separate article (needs more paper).

1. Introduction

Question (P vs. NP)

"If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem?"

Two possibilities of answer are correct for this question.

Answer 1
Not. It's not a truth.
Problem of class P is different than problem of class NP (P≠NP). For classical mathematics it's a correct answer.

Answer 2
Yes. It's a truth.
If problem possess a special algorithmic nature, then (P=NP). Problem of class P is equivalent with the NP problem. For special informatics problems, it's a correct answer (but not for a classical mathematics).

We should focus only on the second solution, where (P=NP). Only at the informatics area, we have a chance to solve this problem. In area of classical mathematics it's a mission impossible.

General speaking, the (P=NP) it is a class of problems which possess algorithmic nature. Correct and fast algorithm is the answer for the prime question. It's a specific class of algorithms (not every kind of algorithms). Algorithm for the problem (P=NP) should be correct/proper and fast. Algorithm should be faster than classical mathematical solution of problem (if classical mathematical solution exists). Proper meaning is that the algorithm should be more effective than standard recurrence. It's possible if algorithm contain logical functions. For example:
– instruction (if…then),
– Boolean operations.

2. Mathematical formula for problem (P=NP)

At first, we should choose the correct question for the problem.

The answer for the problem (P=NP) must be always the algorithm (proper and fast).

$$\overset{\uparrow OK}{Q} \Leftrightarrow \overset{\uparrow OK}{A}_{nswer}$$

$\overset{\uparrow OK}{Q}$ – Proper question for the problem. Not all of possible questions are correct. Proper questions have status – OK.

$\overset{\uparrow OK}{A}_{nswer}$ – Proper answer for the question. The answer for problem of class (P=NP) must be always the algorithm. The algorithm should be correct and fast. Numerical result is not the proper answer. Only correct algorithm is the right answer for the problem!

Correct answer/algorithm have status – OK.

Problem components

- **First step**
  We should have a problem with some correct function/equation for this problem. Possible is use of several equations for the problem description, obviously.

- **Second step**
  We should find a proper and fast algorithm for equation. It's very good for us, if the algorithm contain a logical operation (if….then, or Boolean functions).

- **Third step.**
  Solution time of algorithm should be smaller than time for classical solution, if that classical solution exists:
  - For some class of problems, classical solution exists and algorithm exists as well,
  - For some class of problems, classical solution not exists, but algorithm exists,
  - For some class of problems, classical solution exists, but we need infinity time to solving this problem. It's impossible in real world. We should conclude that the classical solution is extremely difficult, but algorithm for this problem exists and is quite easy.

Formula for the problem with only one function/equation

$$(Pvs.NP) = \overset{\uparrow FD}{f}(x_K) \times \overset{\downarrow AD}{A}(x_K)$$

$$FD > AD$$

(P vs.NP) problem is presented as a function/equation for this problem $f(x_K)$ and a proper algorithm $A(x_K)$. The algorithm is very helpful to quickly find the arguments of function (x_K). For algorithm, arguments of function are called as variables.

Function
Solution of function is always easy to check, if arguments of function are known (x_K) (variables are known).

Algorithm
Proper algorithm is the essential of the problem (P=NP). Without the algorithm, we have a classical mathematical problem. If we wrote correct and fast algorithm for the problem solution, then it's a problem of class (P= NP).

(P vs.NP)
"If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem?"

For the algorithm, sentence ***("is easy to check")*** is the same what a sentence ***("is easy to solve").*** Number of iterations are not changing for the algorithm. **(P=NP)** if we use the proper and fast algorithm for the problem solution.

sentence 1 = sentence 2

("*solve the problem*") = ("*check that a solution to a problem is correct*")

The right answer for the problem's (P=NP) must be always algorithm. Numerical results are not the proper answer.

$\uparrow FD$ – function difficulty index (How many time/steps to solution of problem?)

If we use a computer to solve a classical function, **FD** index is a steps of recurrence. It's also possible to use a computer's time as **FD** index.

$\downarrow AD$ – algorithm difficulty index (How many time/steps to solution of problem?)

It's a sum of iteration steps for algorithm. It's also possible to use a computer's time as **AD** index. **AD** should be smaller than **FD** for problems (P=NP).

FD > AD	**–**	**Problems should be classified as (P=NP)**
		Effective algorithm. It's easy to solve and is also easy to check.
FD = AD	–	Classical mathematical problems.
		Algorithm is the recurrence. Problems shouldn't be classified as (P=NP).
FD < AD	–	Not effective algorithm, difficult to solve.
		Problems can't be classified as (P=NP).

Formula for the problem with several equations:

It's possible, that two equations (or more) are necessary for complete solution of problem. If it's a truth, we should improve a little the formula.

$$(Pvs.NP) = \left[\sum_{K=1}^{E} \overset{\uparrow FD_k}{f_K} (x_K) \right]^{\uparrow FD\_MAX} \times \overset{\downarrow AD}{\underset{K}{A}} (x_K)$$

$$FD\_MAX > AD$$

$\uparrow FD\_MAX$ – sum of difficulty index (maximal value).

E – How many equations exists?

For example. If we have two equations, formula should have a next form:

E=2 – two equations for the problem

K=1 – first equation

K=2 – second equation

$$(Pvs.NP) = \begin{bmatrix} (K=1) & \overset{\uparrow FD_1}{f_1}(x_1) \\ \\ (K=2) & \overset{\uparrow FD_2}{f_2}(x_2) \end{bmatrix}^{\uparrow FD\_MAX} \times \overset{\downarrow AD}{\underset{\frac{1}{2}}{A}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$FD\_MAX = FD_1 + FD_2$$

$$FD\_MAX > AD$$

---

3. Example of problem (P=NP). Version A (easy)

I give you the function. I give you also correct solution for the first 20 points (function and b parameter).

$y = 2 \cdot x^n$ – function

$n = \cos(b)$

$b = ?$ – b parameter

Question

Could you give me next 20 correct points for the function and b parameter?

Values for b parameter (b1): Values for function (y1):

b1 =

	0
0	1
1	2.908554
2	1.997482
3	2.250851
4	1.636212
5	2.910735
6	0.875699
7	6.634041
8	104.452136
9	40.619772
10	7.828661
11	16.674026
12	7.757926
13	19.984752
14	125.310022
15	3377.571049
16	475.997966
17	1089.45279
18	220.700078
19	3646.331027
20	1637.702068
21	

y1 =

	0
0	0
1	2.908554
2	0.686761
3	1.126844
4	0.72693
5	1.778948
6	0.300851
7	7.575713
8	15.744875
9	0.388884
10	0.19273
11	2.12987
12	0.46527
13	2.576043
14	6.270282
15	26.953718
16	0.140929
17	2.288776
18	0.202579
19	16.521657
20	0.449137
21	

Figure 1. Values for the function (y) and (b) parameter (20 points).

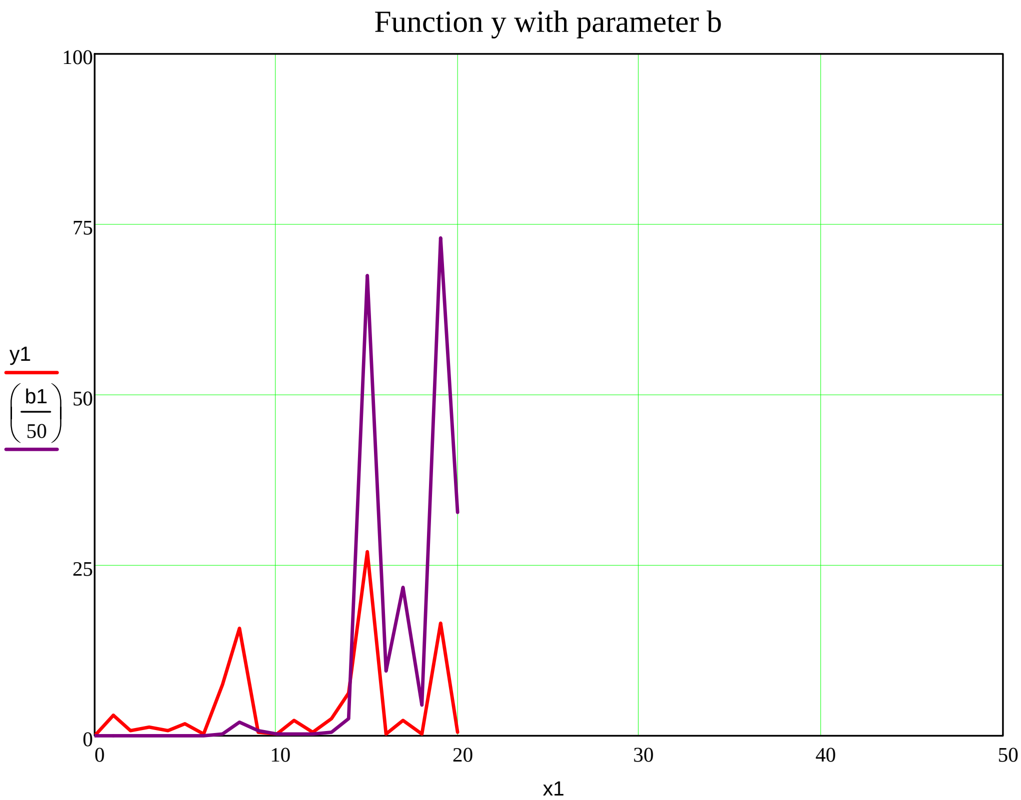

Figure 2. Function and b parameter (20 points). Parameter (b) is divided by 50 for better view.

Answer 1

No, I can’t!

Classical solution by recurrence is possible, but is very difficult. For classical solution, this problem can't be classified as (P=NP).

The truth is, I didn't show the correct answer yet! Why?

Only the algorithm is a right answer for this question. The numerical result is not the correct answer!

Answer 2

Yes, I can! It is algorithm.

Algorithmic solution exists and is very easy. Algorithm if exists, is always easy to check and solve.

sentence 1 = sentence 2
(“*solve the problem”*) = (“*check that a solution to a problem is correct”*).

That is main postulate for problems (P vs. NP).

$y = 2 \cdot x^n$ – function

$n = \cos(b)$

$b_1 = 1$ – start value

$b_i = b_i \cdot \prod_{i=1}^{40} y_i$ – b parameter

Mathcad program/algorithm

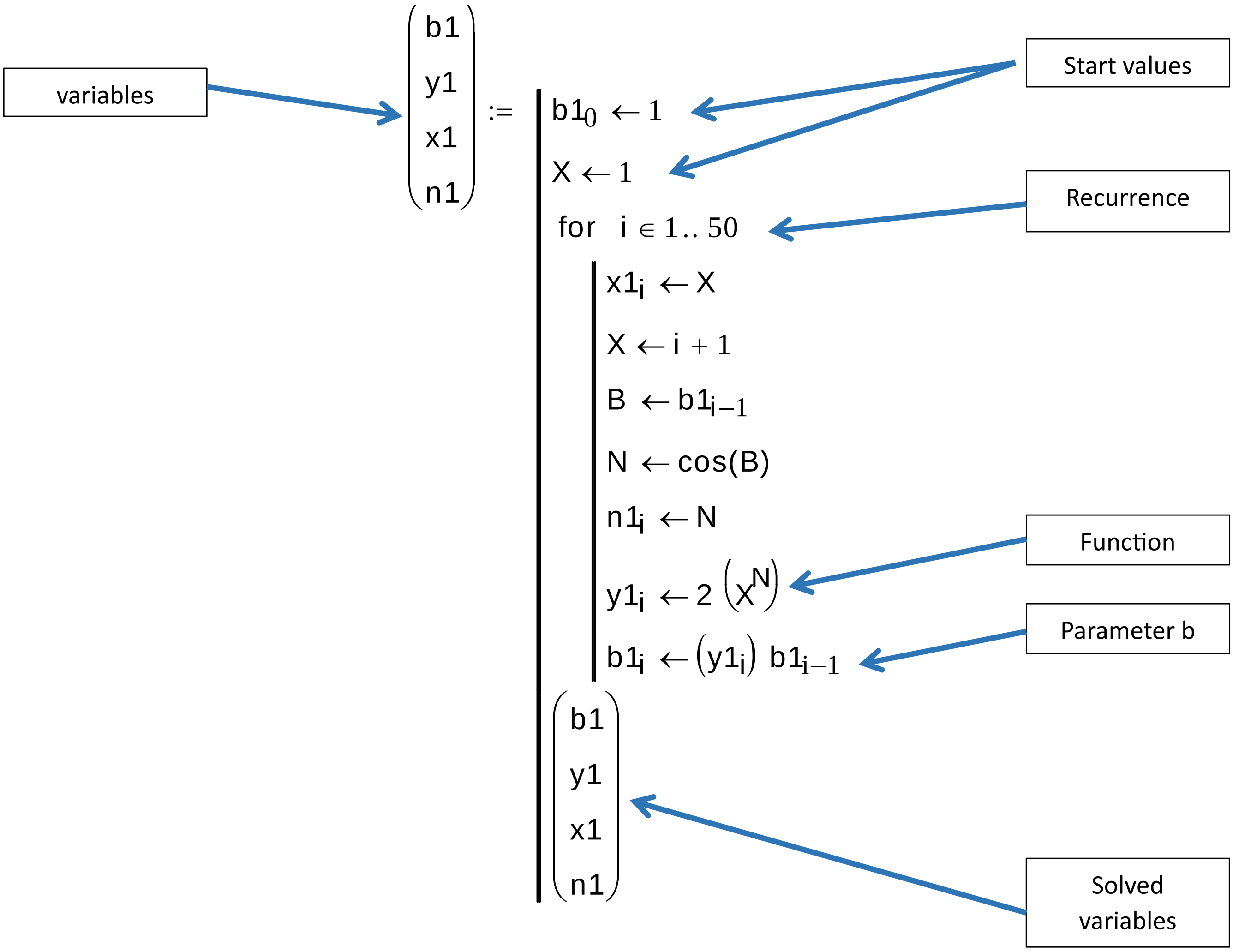

Figure 3. Mathcad program/algorithm.

40 steps of algorithm give the 40 correct points of solution. First 20 points and next 20 correct points for the function and b parameter.

Values for b parameter (b1):

b1 =

	0
20	1637.702068
21	519.028938
22	88.231267
23	3785.100162
24	462.640764
25	101.822555
26	505.064446
27	84.702505
28	5.985934
29	309.388729
30	755.448156
31	2171.706955
32	451.35475
33	5469.470885
34	313.519163
35	11104.592266
36	2622.993596
37	152.73322
38	82.334831
39	3080.790359
40	1189.843379
41	...

Values for function (y1):

y1 =

	0
20	0.449137
21	0.316925
22	0.169993
23	42.89976
24	0.122227
25	0.22009
26	4.960241
27	0.167706
28	0.07067
29	51.685954
30	2.441744
31	2.874727
32	0.207834
33	12.117898
34	0.057322
35	35.419182
36	0.236208
37	0.058229
38	0.539076
39	37.417826
40	0.386214
41	...

Figure 4. Values for the function (y) and (b) parameter (next 20 points).

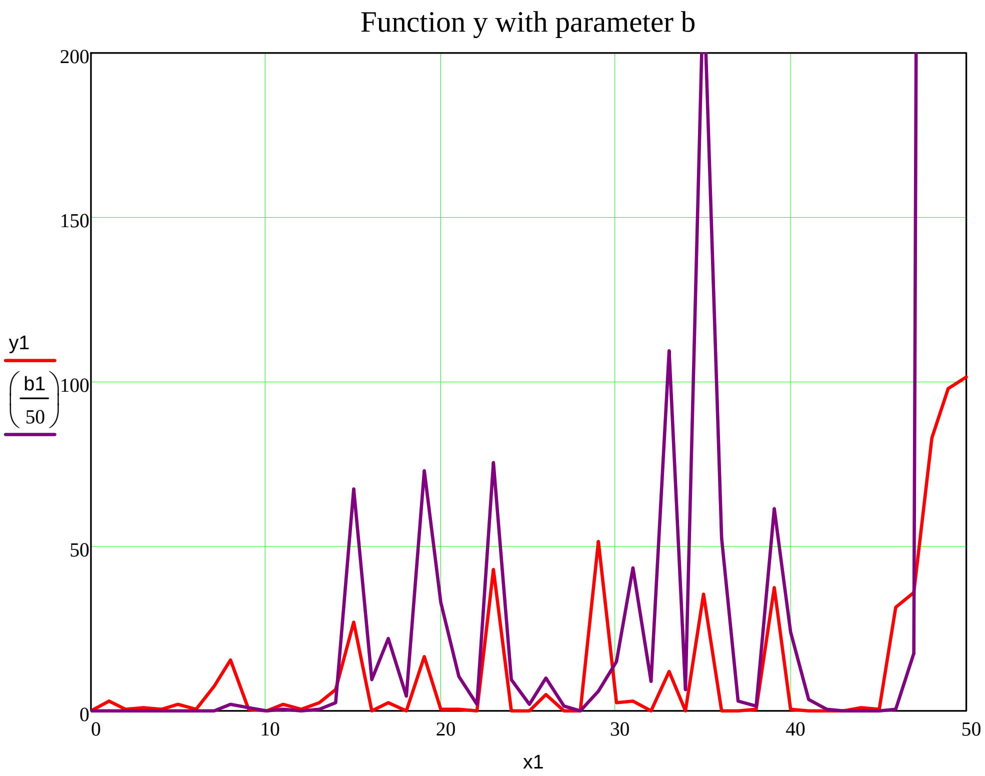

Figure 5. Function and b parameter (50 points). Parameter is divided by 50 for better view.

$\uparrow FD = many \quad steps$

$\downarrow AD = 40$ – Only 40 steps of algorithm.

$$FD > AD$$

$$(Pvs.NP) = \overset{\uparrow FD=many}{f} (x_K) \times \overset{\downarrow AD=40}{A} (x_K)$$

(P vs. NP)

"If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem?"

Sentence 1 - *"it is easy to check that a solution to a problem is correct"* = 40 iteration steps

Sentence 2 - *"it is also easy to solve the problem"* = 40 iteration steps

sentence 1 = sentence 2

---

4. Example of problem (P=NP). Version B (hard)

The same problem as before. I introduced a little modification at algorithm. The logical function was used (if....then). From this moment, it's a very strong version of algorithm, but it is fast algorithm as before. Without the algorithm, solution of problem is impossible. The logical functions are very powerful. The algorithm is a right tool to use them.

Mathcad program/algorithm

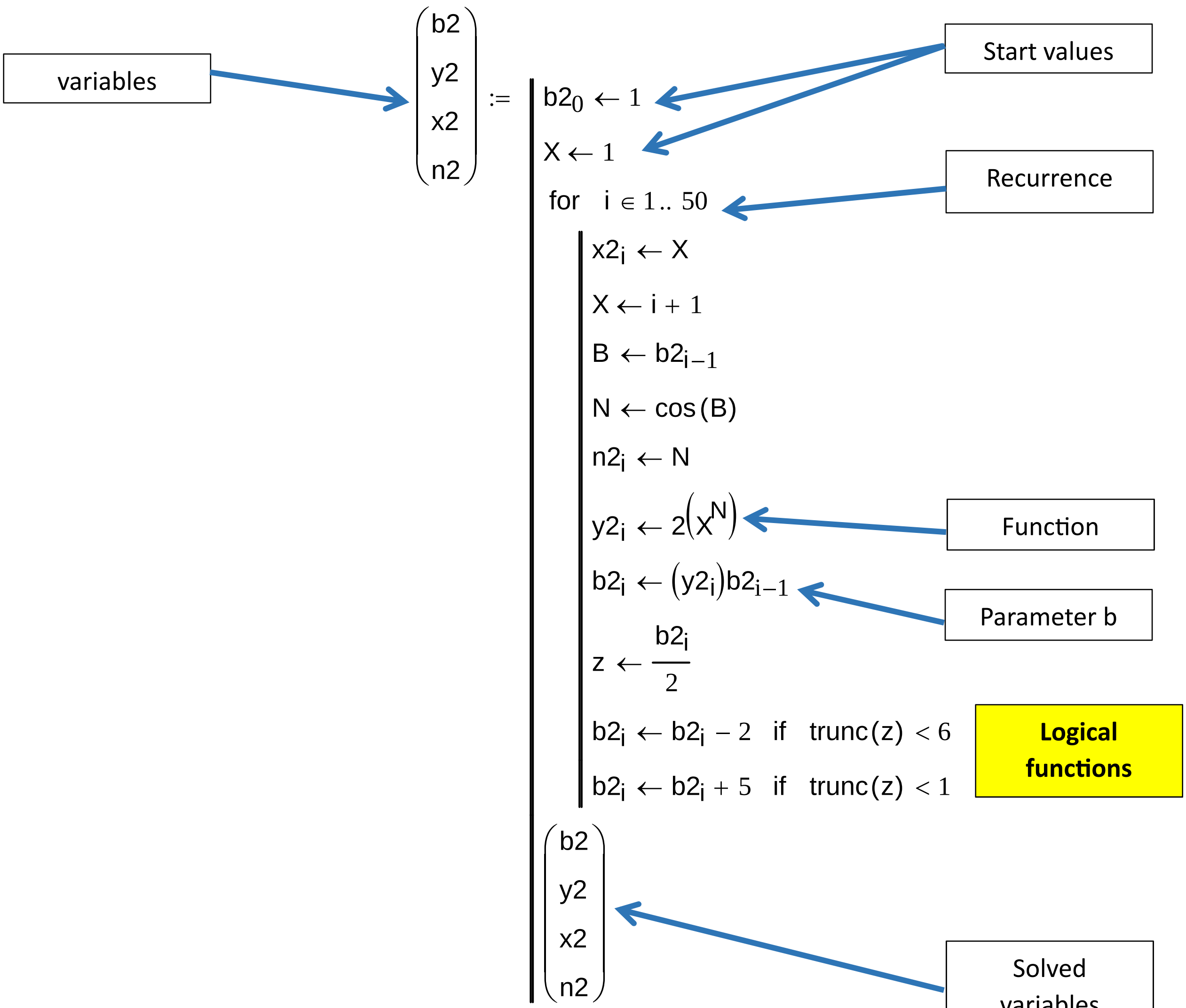

Figure 6. Mathcad program/algorithm. Logical operations were introduced.

Values for b parameter (b2):

b2 =

	0
0	1
1	0.908554
2	1.570724
3	1.141762
4	2.460381
5	4.223366
6	1.386042
7	2.061709
8	4.463517
9	3.062865
10	3.561036
11	3.736162
12	3.892639
13	4.131054
14	4.867463
15	14.938699
16	1.902658
17	4.483948
18	2.603683
19	3.397482
20	3.35729
21	...

Values for function (y2):

y2 =

	0
0	0
1	2.908554
2	3.930116
3	2.0002
4	3.906576
5	0.497226
6	0.80174
7	2.930438
8	0.709856
9	1.134277
10	0.183174
11	0.206727
12	0.238919
13	0.290562
14	0.452055
15	3.069094
16	0.261245
17	0.779935
18	1.026703
19	0.152661
20	0.105163
21	...

Figure 7. Values for the function (y) and (b) parameter (20 points).

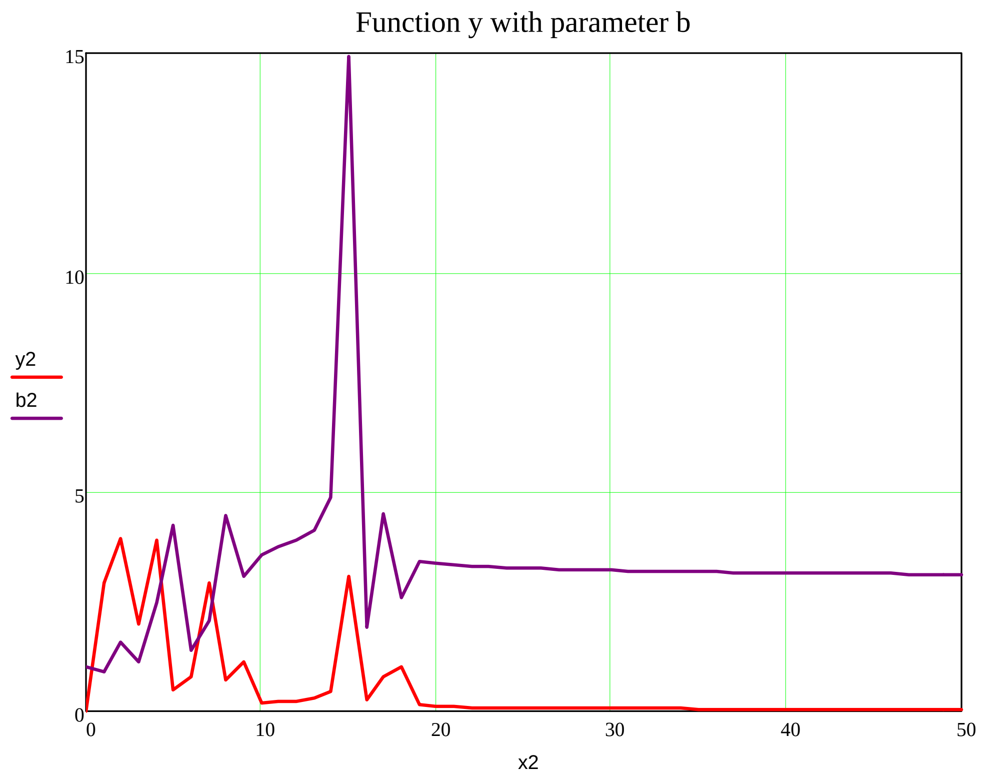

Figure 8. Function and b parameter (50 points). The shape of function (y2) is extremely different than previous function shape (y1).

$\uparrow FD = \inf$ **(infinity time)**

$\downarrow AD = 40$ – Only 40 steps of algorithm.

$$FD > AD$$

$$(Pvs.NP) = \overset{\uparrow FD=\inf}{f}(x_K) \times \overset{\downarrow AD=40}{A}(x_K)$$

(P vs. NP)

"If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem?"

Sentence 1 - *"it is easy to check that a solution to a problem is correct"* = 40 iteration steps

Sentence 2 - *"it is also easy to solve the problem"* = 40 iteration steps

sentence 1 = sentence 2

Answer

Algorithmic solution exists and is very easy. Algorithm is very easy to solve and check. That is main postulate for the problems (P vs. NP).

Conclusion

Final answer for the problem (P vs. NP) sounds YES. It's possible to find a problem of class (P=NP). Exists a lot of problems of that class, probably millions.

"Algorithms with a logical operation are the gate to the new area beyond mathematics".

author