MINISTRY OF EDUCATION OF REPUBLIC OF BELARUS
ESTABLISHMENT OF EDUCATION
"BREST STATE TECHNICHNICAL UNIVERSITY"

Practice work №2
« Minimization of Test Functions»

Made by:
Ivan Bakunovich
Checked by:
Prof. Akira

2016

Task

1. Minimize the following y in (i) 20-D and (iii) 2-D cases.

$$y = nA + \sum_{i=1}^{n} (x_i^2 - Acos(2\pi x_i))$$

2. Show the following graphics in each of 3 cases (i), (ii) and (iii).
(1) the graph of _tness vs generation.
(2) Create a population of 20 chromosomes at random, with _tness being y.

Part 1. 20-D

Program code (C#)

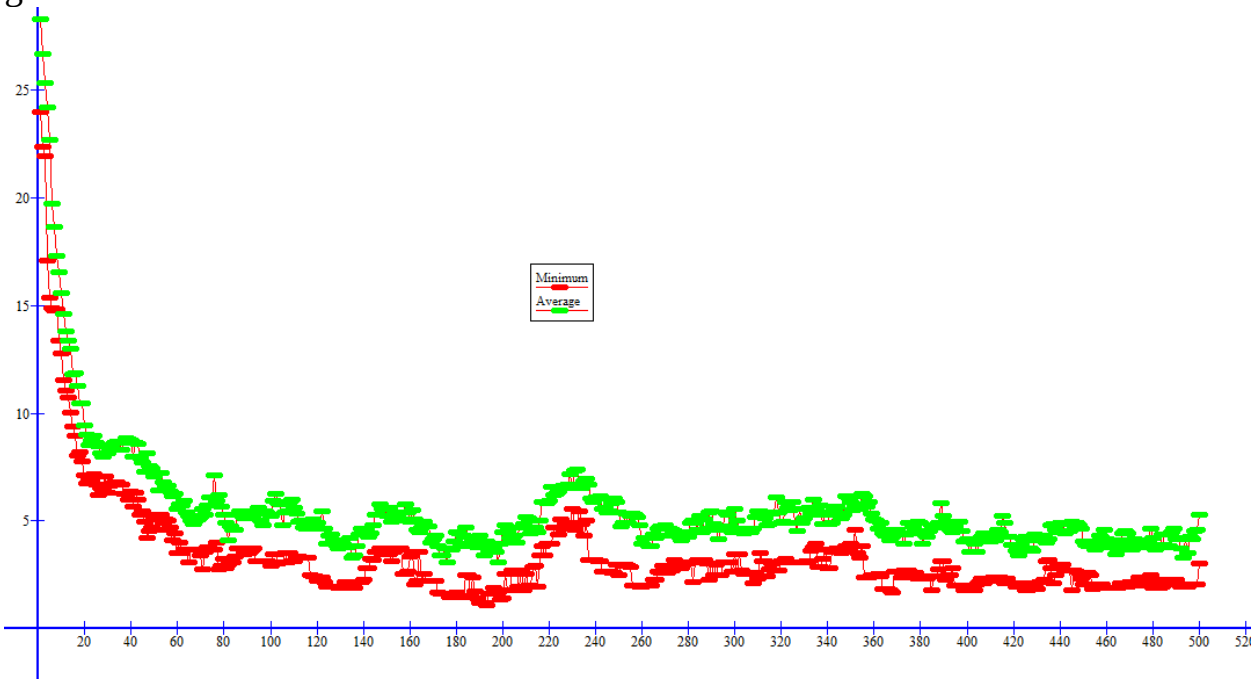
```
namespace CIIT_Lab3
{
    class Work
    {
        public double[,] genes;
        public List<double[]> y;
        public List<double> average_generation_gen;
        public List<double> max_generation_gen;
        public double[] fortime;
        public int[] keys;
        double min, max;
        int size;
        Random rnd;
        public void Generate()
        {
            double[] fortime = new double[size];
            min = -1; max = 1;
            y = new List<double[]>();
            rnd = new Random();
            size = 20;
            genes = new double[size, size];
            for (int i = 0; i < size; i++)
                for (int j = 0; j < size; j++)
                    genes[i, j] = rnd.NextDouble() * (max - min) + min;
                    //genes[i, j] = (double)rnd.Next(-10000, 10001) / 10000;
            max_generation_gen = new List<double>();
            average_generation_gen = new List<double>();
        }
        public void FindY()
        {
            double[] Y = new double[size];
            for (int i = 0; i < size; i++){
                for (int j = 0; j < size; j++)
                    Y[i] += (Math.Pow(genes[i, j], 2) - Math.Cos(2 * Math.PI * genes[i, j]));
                Y[i] += size;
            }
            y.Add(Y);
        }
        public void Delete_Bad_Genes()
        {
            keys = new int[size];
            for (int i = 0; i < size; i++) keys[i] = i;
            fortime = new double[size];
            fortime = y.Last().ToArray();
            Array.Sort(fortime, keys);
```

```
            average_generation_gen.Add(y.Last().Average());
            max_generation_gen.Add(y.Last().Min());
        }
        public void Cross()
        {
            int p = 0;
            double[,] newgenerat = new double[size, size];
            for (int i = 0; i < size/2; i++)
            {
                int first_parent = rnd.Next(0, size / 2);
                int second_parent = rnd.Next(0, size / 2);
                double[] first_parent_mass = new double[size];
                double[] second_parent_mass = new double[size];
                for (int j = 0; j < size; j++)
                {
                    first_parent_mass[j] = genes[keys[first_parent],j];
                    second_parent_mass[j] = genes[keys[second_parent], j];
                }
                int num = rnd.Next(1, size);
                for (int j = num; j < size; j++)
                {
                    double k = first_parent_mass[j];
                    first_parent_mass[j] = second_parent_mass[j];
                    second_parent_mass[j] = k;
                }
                for(int j = 0; j < size; j++)
                {
                    newgenerat[p, j] = first_parent_mass[j];
                    newgenerat[p + 1, j] = second_parent_mass[j];
                }
                p += 2;
            }
            genes = new double[size, size];
            genes = newgenerat;
        }
        public void mutation()
        {
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    int num = rnd.Next(0, size);
                    if (num == 0)
                    {
                        genes[i, j] = rnd.NextDouble() * (max - min) + min;
                    }
                }
            }
        }
        public bool Uslovie()
        {
            if (max_generation_gen.Count == 500) return true;
            return false;
        }
        public void picture()
        {
            Console.WriteLine("Generation " + max_generation_gen.Count);
            Console.WriteLine("Min = " + max_generation_gen.Last());
            Console.WriteLine("Average = " + average_generation_gen.Last());
        }
        public void FTLE()
        {
            StreamWriter file = new StreamWriter(@"B:\All_study_stuff_are_here\4
course\СИИТ\lab3\max.txt", true);
```

```
            StreamWriter file1 = new StreamWriter(@"B:\All_study_stuff_are_here\4
course\СИИТ\lab3\avg.txt", true);
            StreamWriter file2 = new StreamWriter(@"B:\All_study_stuff_are_here\4
course\СИИТ\lab3\num.txt", true);
            file.WriteLine(max_generation_gen.Last().ToString());
            file1.WriteLine(average_generation_gen.Last().ToString());
            file2.WriteLine(max_generation_gen.Count().ToString());
            file.Close();
            file1.Close();
            file2.Close();
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Work obj = new Work();
            obj.Generate();
            do
            {
                obj.FindY();
                obj.Delete_Bad_Genes();
                obj.picture();
                obj.FTLE();
                obj.Cross();
                obj.mutation();
            } while (obj.Uslovie() == false);
            Console.WriteLine("An excellent gene!\n");
        }
    }
}
```

RESULTS

Graphic of best chromosomes (red graphic) and average (green graphic) in all generations:



Part 1. 2-D

Program code (C#)

```
namespace CIIT_Lab3
{
    class Work1
    {
        public double[] fortime;
        public int[] keys;
        public List<double[]> y;
        public int[,] genes;
        public List<double> average_generation_gen;
        public List<double> max_generation_gen;
        Random rnd;
        public void Generate()
        {
            y = new List<double[]>();
            rnd = new Random();
            genes = new int[20, 11];
            for (int i = 0; i < 20; i++)
                for (int j = 0; j < 11; j++)
                    genes[i, j] = rnd.Next(0,2);
            max_generation_gen = new List<double>();
            average_generation_gen = new List<double>();
        }
        public void FindY()
        {
            double[] Y = new double[20];
            for(int i = 0; i < 20; i++)
            {
                String s = "";
                for(int j = 0; j< 11; j++)
                {
                    if (j == 0) continue;
                    else
                    {
                        s += genes[i, j].ToString();
                    }
                }
                Y[i] = Convert.ToInt32(s, 2);
                Y[i] /= 1023;
                if (genes[i, 0] == 0) Y[i] = -Y[i];
            }
            for (int i = 0; i < 20; i++)
                Y[i] = 1 + (Math.Pow(Y[i], 2) - Math.Cos(2 * Math.PI * Y[i]));
            y.Add(Y);
        }
        public void Delete_Bad_Genes()
        {
            keys = new int[20];
            for (int i = 0; i < 20; i++) keys[i] = i;
            fortime = new double[20];
            fortime = y.Last().ToArray();
            Array.Sort(fortime, keys);
            average_generation_gen.Add(y.Last().Average());
            max_generation_gen.Add(y.Last().Min());
        }
        public void Cross()
        {
            int p = 0;
            int[,] newgenerat = new int[20, 11];
            for (int i = 0; i < 10; i++)
            {
                int first_parent = rnd.Next(0, 10);
                int second_parent = rnd.Next(0, 10);
                int[] first_parent_mass = new int[11];
                int[] second_parent_mass = new int[11];
                for (int j = 0; j < 11; j++)
```

```
                {
                    first_parent_mass[j] = genes[keys[first_parent], j];
                    second_parent_mass[j] = genes[keys[second_parent], j];
                }
                int num = rnd.Next(1, 11);
                for (int j = num; j < 11; j++)
                {
                    int k = first_parent_mass[j];
                    first_parent_mass[j] = second_parent_mass[j];
                    second_parent_mass[j] = k;
                }
                for (int j = 0; j < 11; j++)
                {
                    newgenerat[p, j] = first_parent_mass[j];
                    newgenerat[p + 1, j] = second_parent_mass[j];
                }
                p += 2;
            }
            genes = new int[20, 11];
            genes = newgenerat;
        }
        public void mutation()
        {
            for (int i = 0; i < 20; i++)
            {
                for (int j = 0; j < 11; j++)
                {
                    int num = rnd.Next(0, 11);
                    if (num == 0)
                    {
                        if (genes[i, j] == 1) genes[i, j] = 0;
                        else genes[i, j] = 1;
                    }
                }
            }
        }
        public bool Uslovie()
        {
            if (max_generation_gen.Count == 500) return true;
            return false;
        }
        public void picture()
        {
            Console.WriteLine("Generation " + max_generation_gen.Count);
            Console.WriteLine("Min = " + max_generation_gen.Last());
            Console.WriteLine("Average = " + average_generation_gen.Last());
        }
        public void FTLE()
        {
            StreamWriter file = new StreamWriter(@"B:\All_study_stuff_are_here\4
course\СИИТ\lab3\max1.txt", true);
            StreamWriter file1 = new StreamWriter(@"B:\All_study_stuff_are_here\4
course\СИИТ\lab3\avg1.txt", true);
            file.WriteLine(max_generation_gen.Last().ToString());
            file1.WriteLine(average_generation_gen.Last().ToString());
            file.Close();
            file1.Close();
        }
}
class Program
    {
        static void Main(string[] args)
        {
            Work1 obj = new Work1();
            obj.Generate();
```

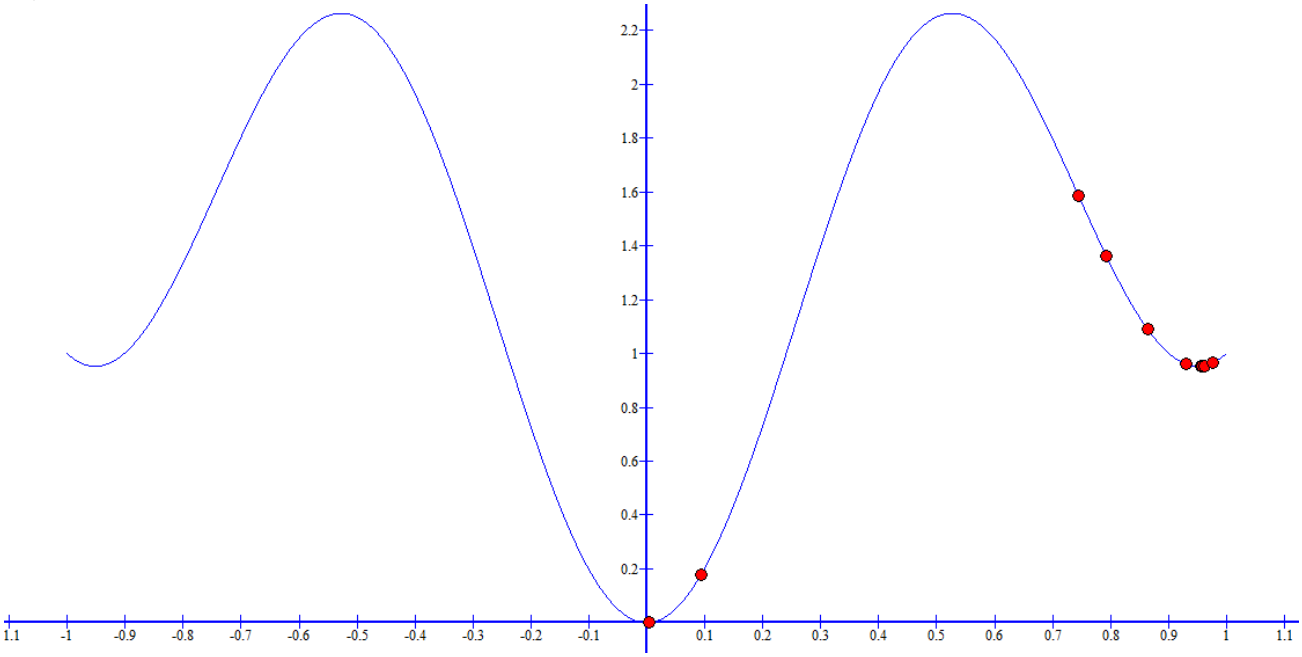
```
            do
            {
                obj.FindY();
                obj.Delete_Bad_Genes();
                obj.picture();
                obj.FTLE();
                obj.Cross();
                obj.mutation();
            } while (obj.Uslovie() == false);
            Console.WriteLine("An excellent gene!\n");
        }
    }
}
```

RESULTS

5 graphics of 2 different generations (first, 3 from average epochs and last generations):

Y – X (in binary):

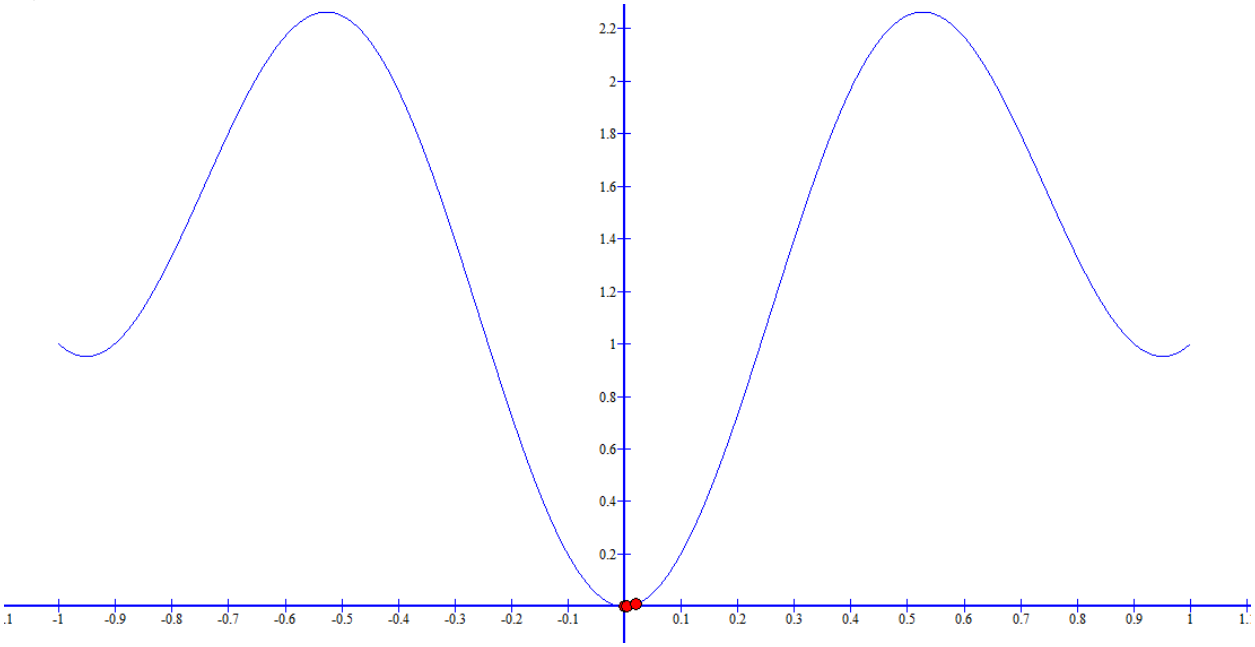
0,000317059016806343 - 0000000100
0,177656679509898 - 0001100000
0,951910641925863 - 1111010010
0,952373174256265 - 1111010100
0,953758621913402 - 1111011000
0,960386486899429 - 1110110111
0,964474192072199 - 1111100111
1,0895337072928 - 1101110100
1,36299148898277 - 1100101011
1,58706788824567 - 1011111010



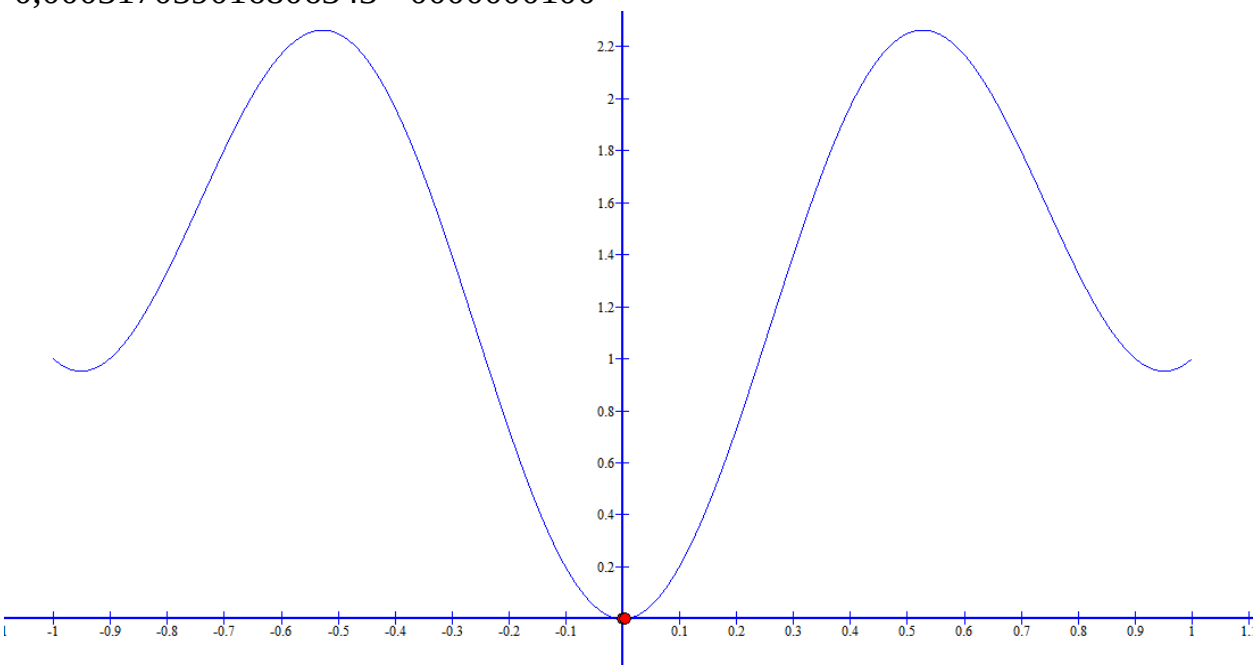
Y – X (in binary):

1,98170779310836E-05 - 0000000001
1,98170779310836E-05 - 0000000001
0,000317059016806343 - 0000000100
0,00049539137345278 - 0000000101
0,00049539137345278 - 0000000101
0,00049539137345278 - 0000000101
0,00049539137345278 - 0000000101

0,00049539137345278 - 0000000101
0,00049539137345278 - 0000000101
0,00872783248626696 - 0000010101

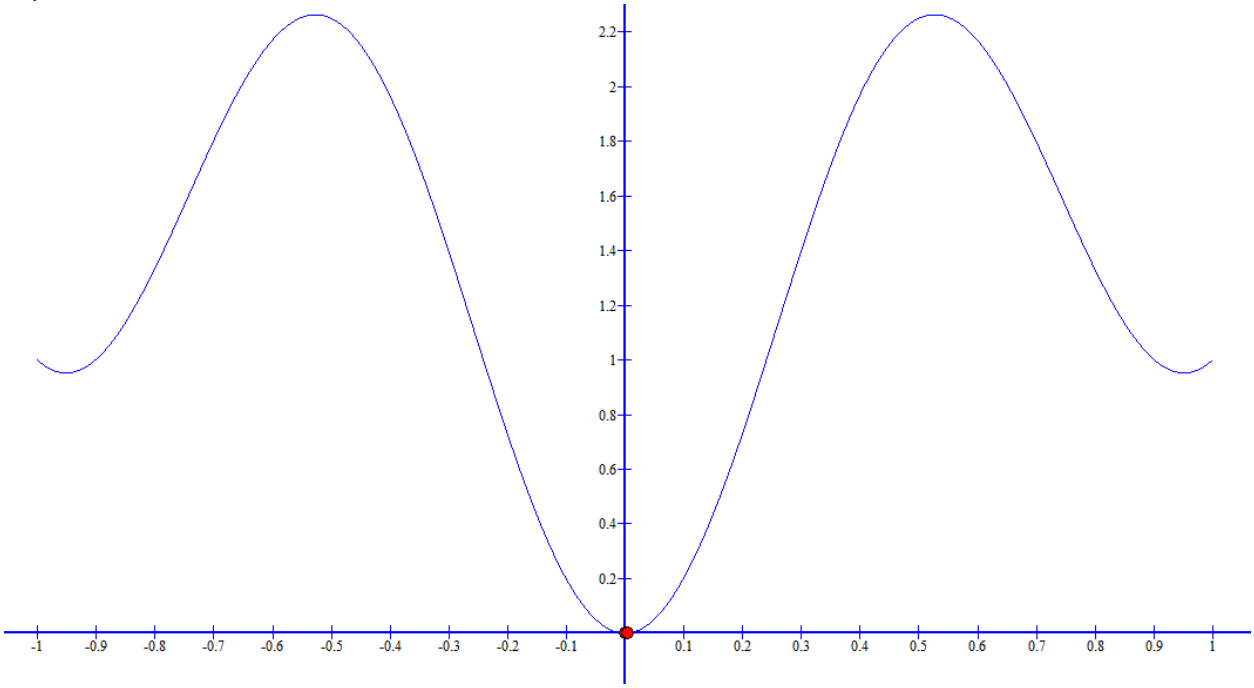


Y – X (in binary):
0 - 0000000000
0 - 0000000000
0 - 0000000000
0 - 0000000000
0 - 0000000000
0,00031705901680634 3 - 0000000100
0,00031705901680634 3 - 0000000100
0,00031705901680634 3 - 0000000100
0,00031705901680634 3 - 0000000100
0,00031705901680634 3 - 0000000100

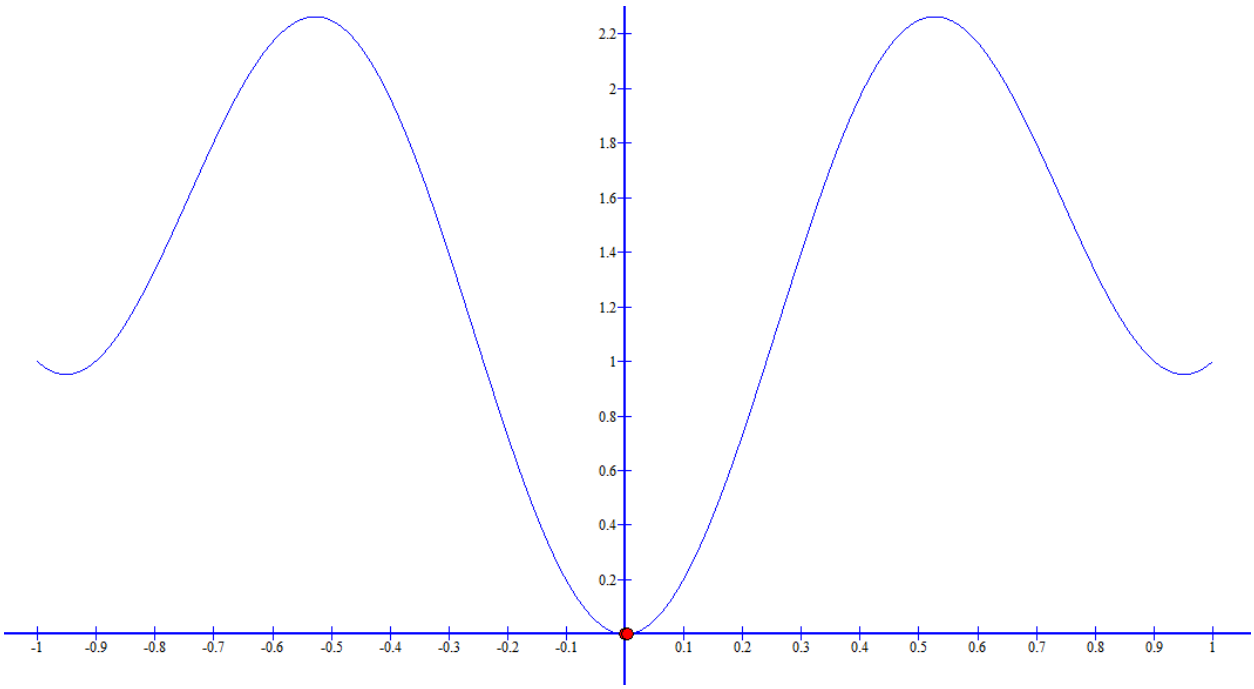


Y – X (in binary):

0 - 0000000000
0 - 0000000000
0 - 0000000000
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
0,000178349432314651 - 0000000011
0,000317059016806343 - 0000000100
0,000713340099422144 - 0000000110



Y – X (in binary):
0 - 0000000000
0 - 0000000000
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
7,92676002088211E-05 - 0000000010
0,000178349432314651 - 0000000011
0,000317059016806343 - 0000000100



And graphic of best chromosomes (red graphic) and average (green graphic) in all generations: