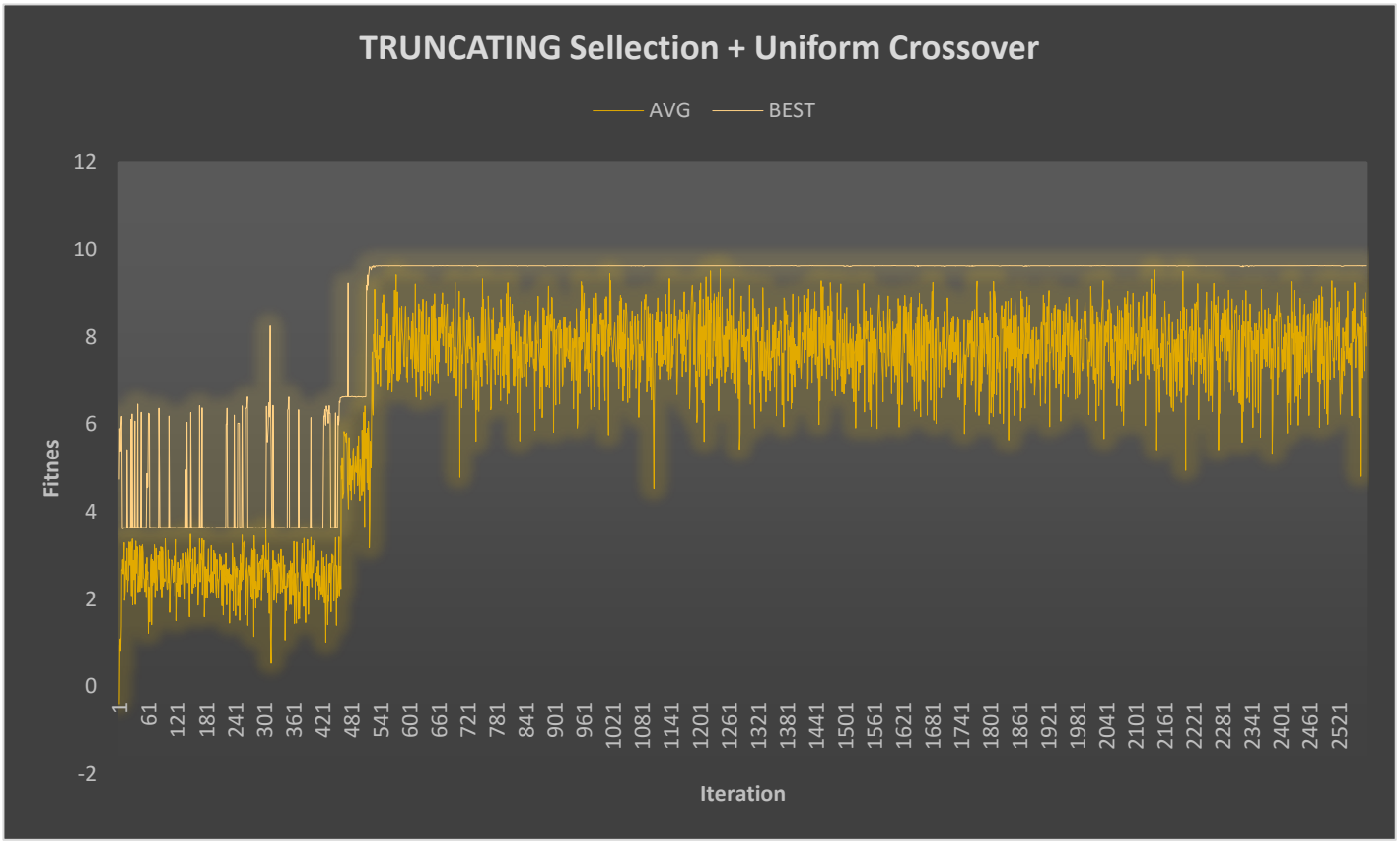
Student: Kirill Tsibikov

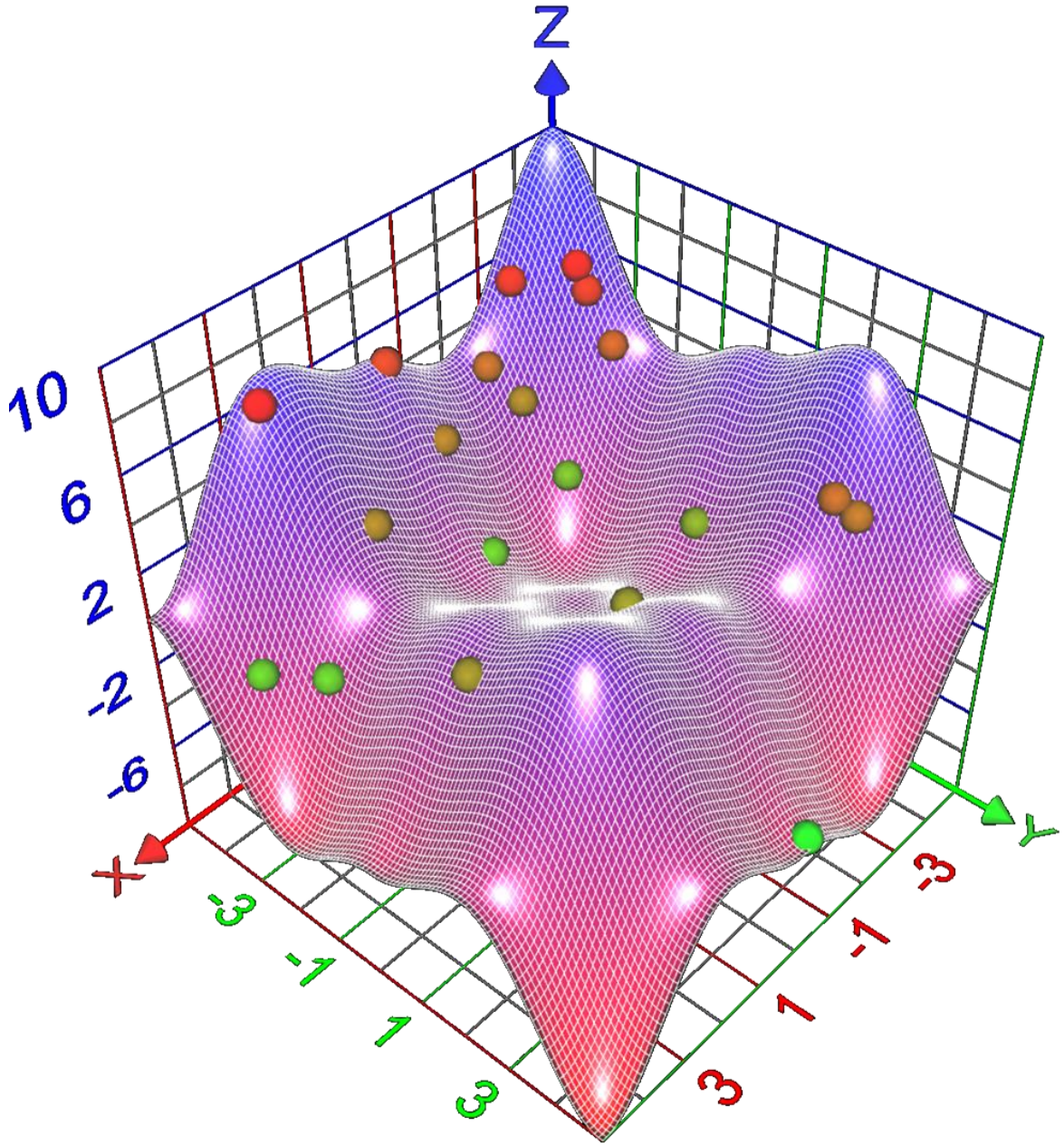
2. Maximization of 3-D Schefel function

Brest 2016

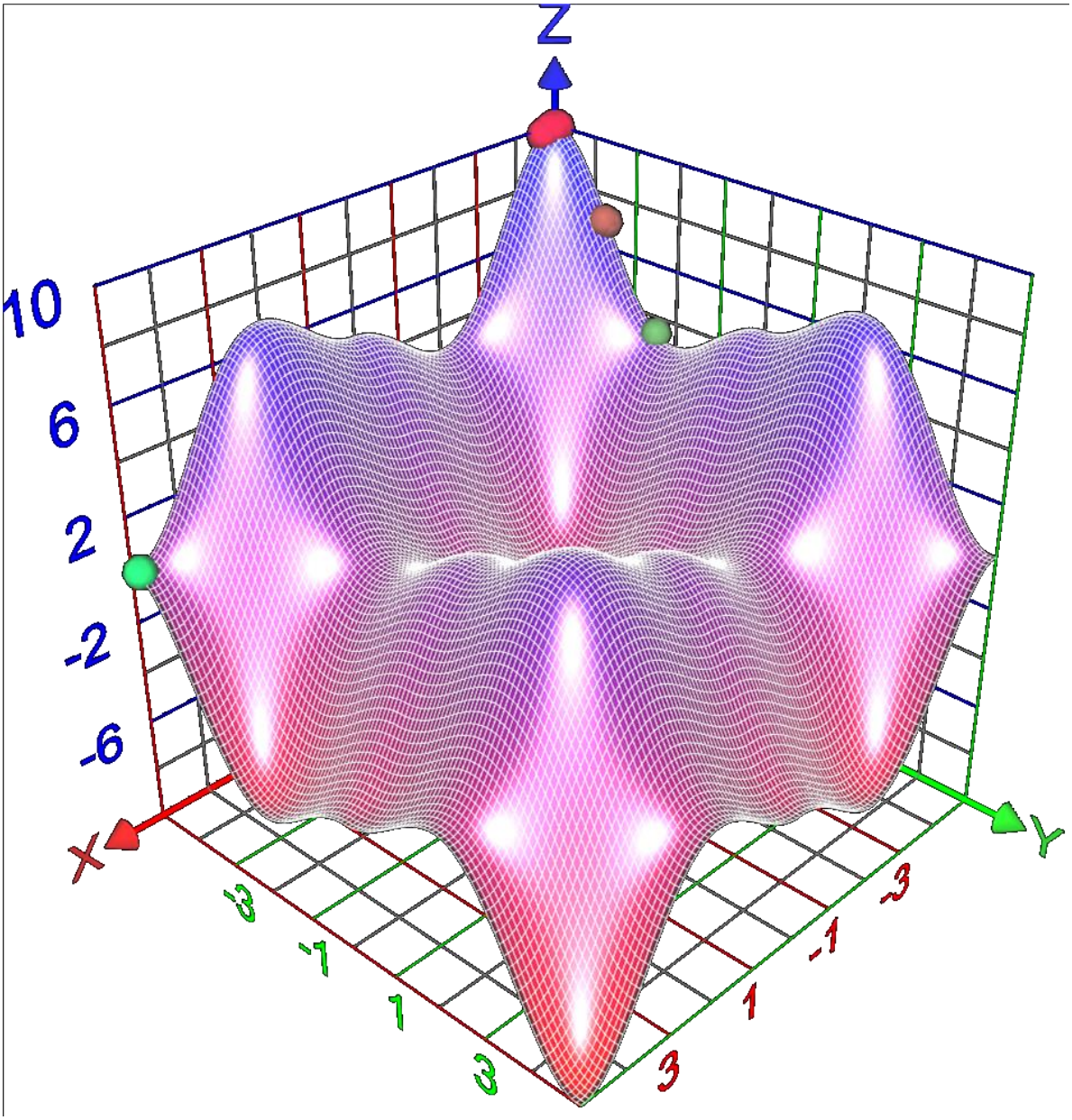
Function: y=x1*sin(abs(x1))+x2*sin(abs(x2))



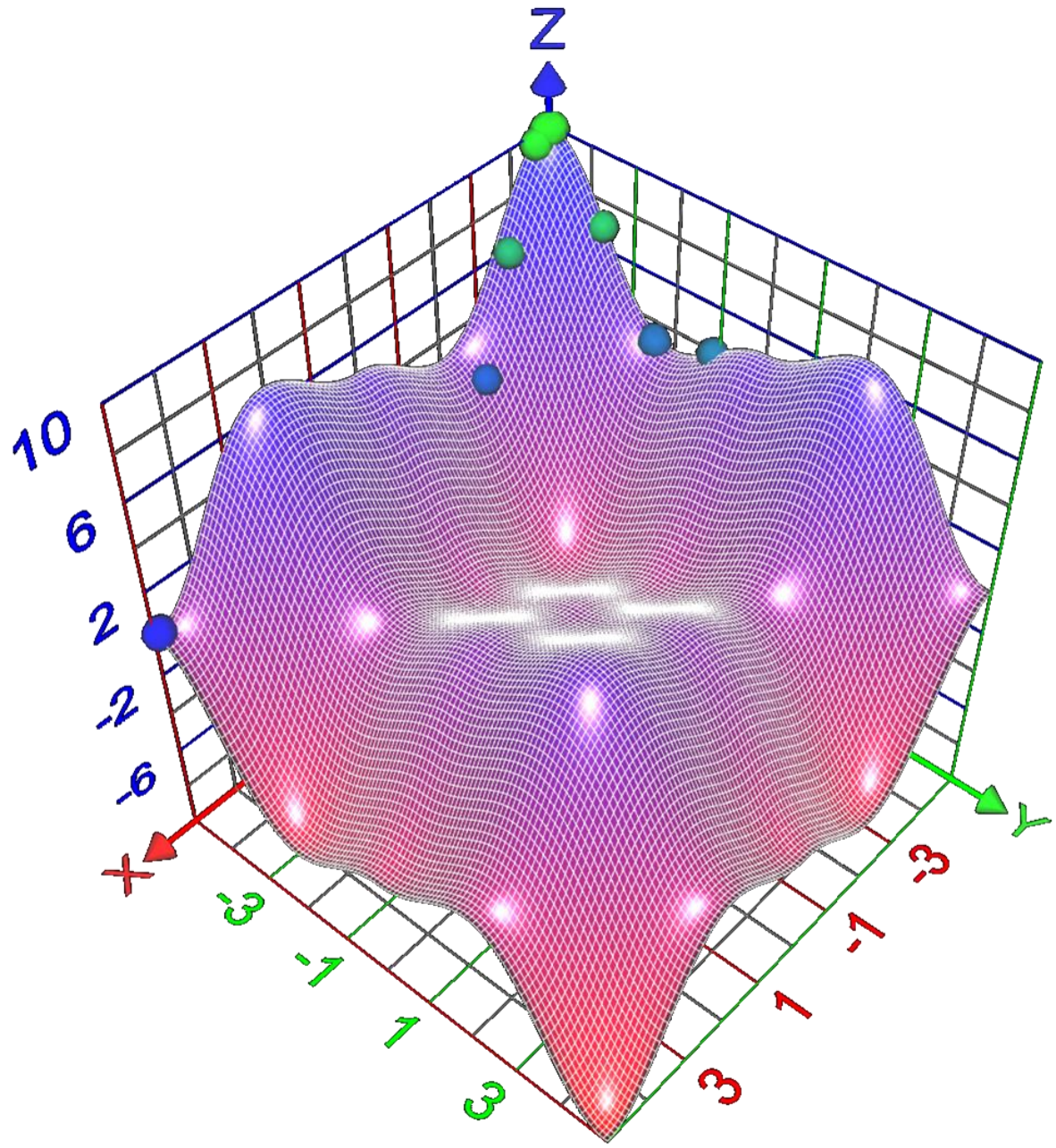
The 1 generation



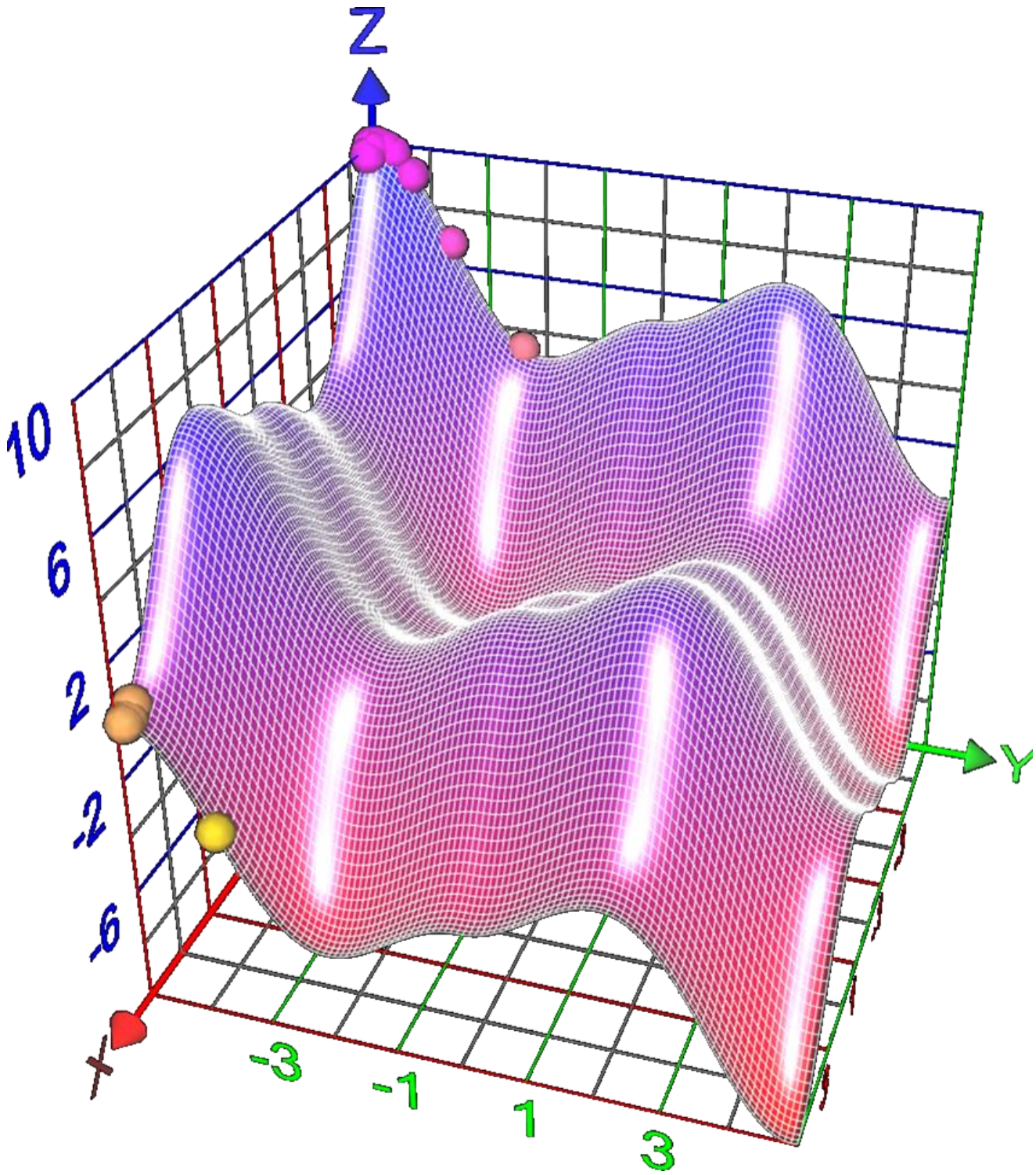
The 50 generation



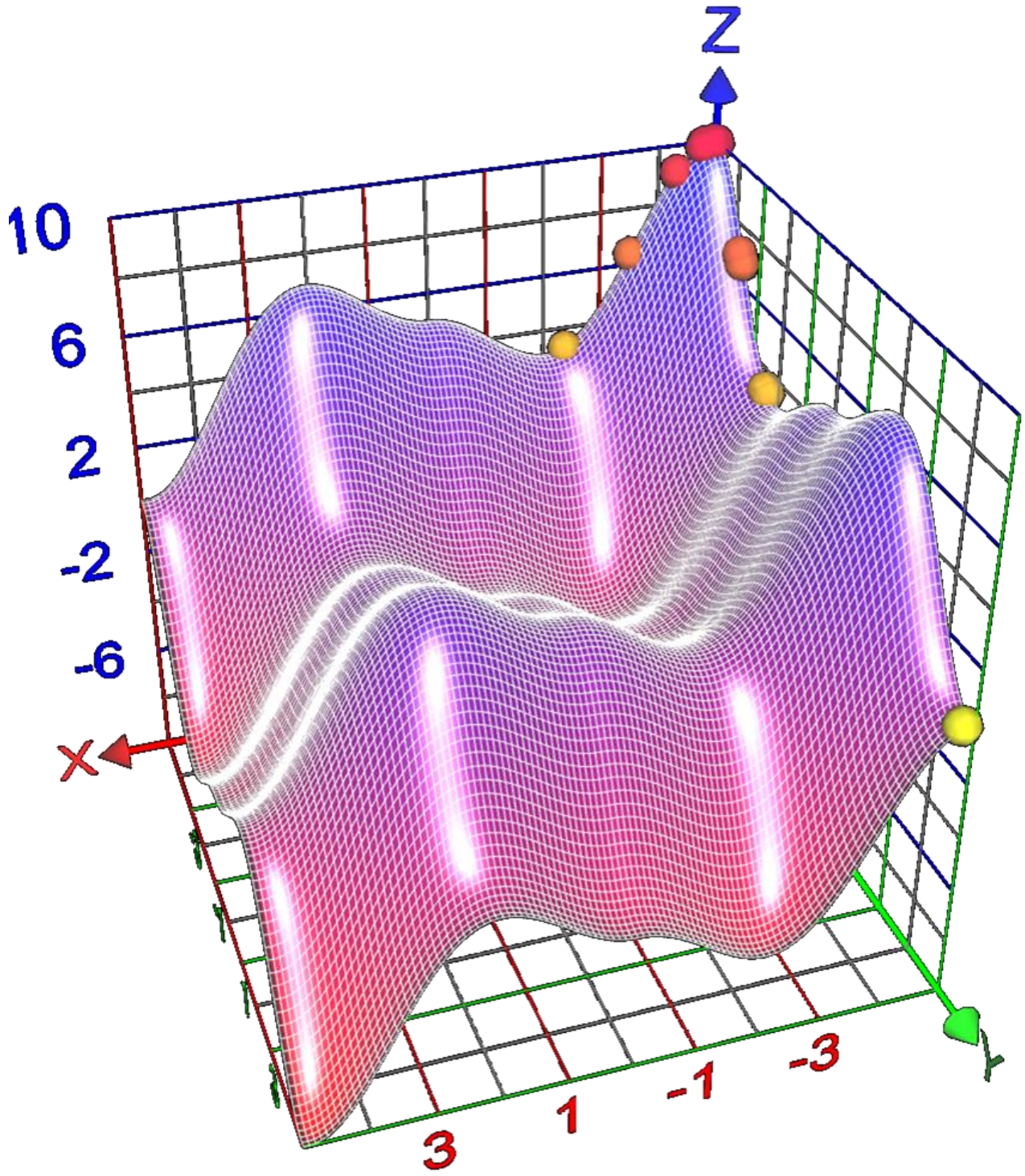
The 200 generation



The 400 generation



The 100000 generation



Listing

```
package javaapplication1;

import java.util.Map;

import java.util.Random;

class GeneticAlgo{

    boolean[][] popln;

    public enum SelectionType {

      TOURNEY, ROULETTE_WHEEL, TRUNCATING

}

    public enum CrossingType {

      ONE_POINT_RECOMBINATION, TWO_POINT_RECOMBINATION,
ELEMENTWISE_RECOMBINATION, ONE_ELEMENT_EXCHANGE

}

    private SelectionType slctp;

    private CrossingType crstp;

    private int genomLength; //Длина генома в битах

    private int generationCount; //Кол-во поколений

    private int individualCount; //Кол-во Геномов(Индивидов,Особей) в
поколении

    private int[] chosenchromosom;

    private SelectionType selectionType; //Тип Селекции

    private CrossingType crossingType; //Тип Скрещивания

    public GeneticAlgo(String s, String c){

        slctp=SelectionType.valueOf(s);

        crstp =CrossingType.valueOf(c);

        popln=new  boolean[20][22];

        genomLength=22;

        generationCount=20;

        individualCount=20;

        chosenchromosom=new int[20];

    }

    public boolean[][] run(){

        this.generateFirstGeneration();

        double ftnsmax=0.D;

        for(int i=0; i<100000000; i++)

        {
```

```
            this.selection();
            ftnsmax=fitnes(0);
            for(int j=1; j<20; j++)
            {
                ftnsmax=(ftnsmax<=fitnes(j))?fitnes(j):ftnsmax;
            }
//            System.out.print(avgfitnes());
//            System.out.print(":");
//            System.out.print(ftnsmax);
//        System.out.println();
            if(i==0||i==50||i==200||i==400||i==100000)
            {
                System.out.println("The " + i + " population");
                for(int j=0; j<20; j++)
                {
                    System.out.print(this.x1(j));
                    System.out.print(":");
                    System.out.print(this.x2(j));
                    System.out.print(":");
                    System.out.print(this.fitnes(j));
                    System.out.println();
                }
            }
            if(ftnsmax==1000)break;
            ftnsmax=0.D;
        }

        return (new boolean[20][22]);
    }
    private void generateFirstGeneration() {
    Random rnd=new Random();
        for(int i=0; i<20; i++)
        {
            for(int j=0; j<22; j++)
            {
                popln[i][j]=rnd.nextBoolean();
```

```
            }
        }
    } //генерация первого поколения
    private void selection(){
        boolean[][] genomListOffsprings=new boolean[20][22];
        Random rndd=new Random();
        switch(this.slctp)
        {
            case ROULETTE_WHEEL:{
                double[] wheel = new double[this.individualCount];
            wheel[0] = fitnes(0);//Значение ФитнессФункции для 1-ого генома
                this.chosenchromosom[0]=0;
            for (int i=1;i<this.individualCount;i++){
                        wheel[i] = wheel[i-1] + fitnes(i);//Значение
ФитнессФункции для i-ого генома
                                this.chosenchromosom[i]=0;
                  }
            double all = wheel[this.individualCount-1];

            for (int i=0;i<this.individualCount;i++){
                    double index = Math.abs(rndd.nextFloat())*all;
                    int l = 0;
                    int r = individualCount-1;
                    int c = 0;
                    while (l < r){
                  c = (l+r) >> 1;
                  if (index <= wheel[c])
                            r = c;
                       else
                     l = c + 1;
                    }
                    int a=l;
                    index = Math.abs(rndd.nextFloat())*all;

                    l = 0;
                    r = individualCount-1;
```

```
        c = 0;
        while (l < r){
      c = (l+r) >> 1;
      if (index <= wheel[c])
              r = c;
          else
       l = c + 1;
        }
        this.chosenchromosom[l]++;
        this.chosenchromosom[a]++;
        genomListOffsprings[i] = this.crossing(l,a);
}
    popln=genomListOffsprings;
    break;
}
case TOURNEY:
{
    for (int i=0;i<this.individualCount;i++){
        int index1 = rndd.nextInt(individualCount);
        int index2 = rndd.nextInt(individualCount);
        int index3 = rndd.nextInt(individualCount);
        int index4 = rndd.nextInt(individualCount);
        double fr1 = fitnes(index1);
        double fr2 = fitnes(index2);
        index1=(fr1>fr2)?index1:index2;
        double fr3 = fitnes(index3);
        double fr4 = fitnes(index4);
        index2=(fr3>fr4)?index3:index4;
        genomListOffsprings[i] = this.crossing(index1, index2);
}
    popln=genomListOffsprings;
    break;
}
case TRUNCATING:
{
    int percent=(Math.abs(rndd.nextInt())+2)%10+1;
```

```
                this.sort();
                for(int i=0; i<this.individualCount; i++)
                {
                    genomListOffsprings[i] =
this.crossing((Math.abs(rndd.nextInt()))%10,(Math.abs(rndd.nextInt()))%10);
                }
                popln=genomListOffsprings;
                break;
            }
            default:
                break;

    }
    } //Процедура селекци
    private boolean[] crossing(int a, int b) {
        boolean[] vec=new boolean[22];
        switch(crstp)
        {
            case ONE_ELEMENT_EXCHANGE:
            {
             for(int i=0; i<genomLength; i++)
             {
                 Random rndd=new Random();
                 vec[i]=(rndd.nextBoolean())?popln[b][i]:popln[a][i];
             }
             break;
            }
            case ONE_POINT_RECOMBINATION:
            {
                Random rndd=new Random();
                int point=Math.abs(rndd.nextInt())%genomLength + 1;
                System.arraycopy(popln[a], 0, vec, 0, point);
                System.arraycopy(popln[b], point-1, vec, point-1,
genomLength-point);
                break;
            }
            default:
```

```
            break;
    }

//mutation
Random rdd=new Random();
for(int i=0; i<20; i++)
{
    vec[i]=(Math.abs(rdd.nextInt())%21==0)?!vec[i]:vec[i];
}
return vec;
} //Процедура скрещивания
private double x1(int nomber){
double a=0.D;
    double i=0.D;
    for(int j=10; j>0; j--, i+=1.D)
        {
            a+=(this.popln[nomber][j])?Math.pow(2.D, i):0;
        }
    a*=(this.popln[nomber][0])?5:-5;
    a/=1023;
    return a;
}
 private double x2(int nomber){
    double i=0.D;
    double b=0.D;
    for(int j=21; j>11; j--, i+=1.D)
        {
            b+=(this.popln[nomber][j])?Math.pow(2, i):0;
        }
    b*=(this.popln[nomber][11])?5:-5;
    b/=1023;
    return b;
}
private double fitnes(int nomber){
    double ftns=0.f;
    double a=0.D;
```

```
    double i=0.D;
    for(int j=10; j>0; j--, i+=1.D)
        {
            a+=(this.popln[nomber][j])?Math.pow(2.D, i):0;
        }
    a*=(this.popln[nomber][0])?5:-5;
    a/=1023;
    i=0.D;
   double b=0.D;
    for(int j=21; j>11; j--, i+=1.D)
        {
            b+=(this.popln[nomber][j])?Math.pow(2, i):0;
        }
    b*=(this.popln[nomber][11])?5:-5;
    b/=1023;
    ftns=a*Math.sin(Math.abs(a))+b*Math.sin(Math.abs(b));
    return ftns;
} //Фитнес функция
private double avgfitnes(){
    double ftns=0.D;
    for(int i=0; i<20; i++)
    ftns+=this.fitnes(i);
    return ftns/20;
} //Фитнес функция
private void sort()
{
    for(int i=0; i<this.individualCount; i++)
    {
        boolean[] amiba;
        amiba = new boolean[22];
        amiba=popln[i];
        double fit=fitnes(i);
        for(int j=i; j<this.individualCount; j++)
        {
            if(fitnes(j)>fit){
                fit=fitnes(j);
```

```
                    amiba=popln[j];
                    popln[j]=popln[i];
                    popln[i]=amiba;
                }
            }
        }
    }
}
public class JavaApplication1 {
    public static void main(String[] args) {
        GeneticAlgo p=new GeneticAlgo("TRUNCATING","ONE_ELEMENT_EXCHANGE" );
        p.run();
      }
}
```