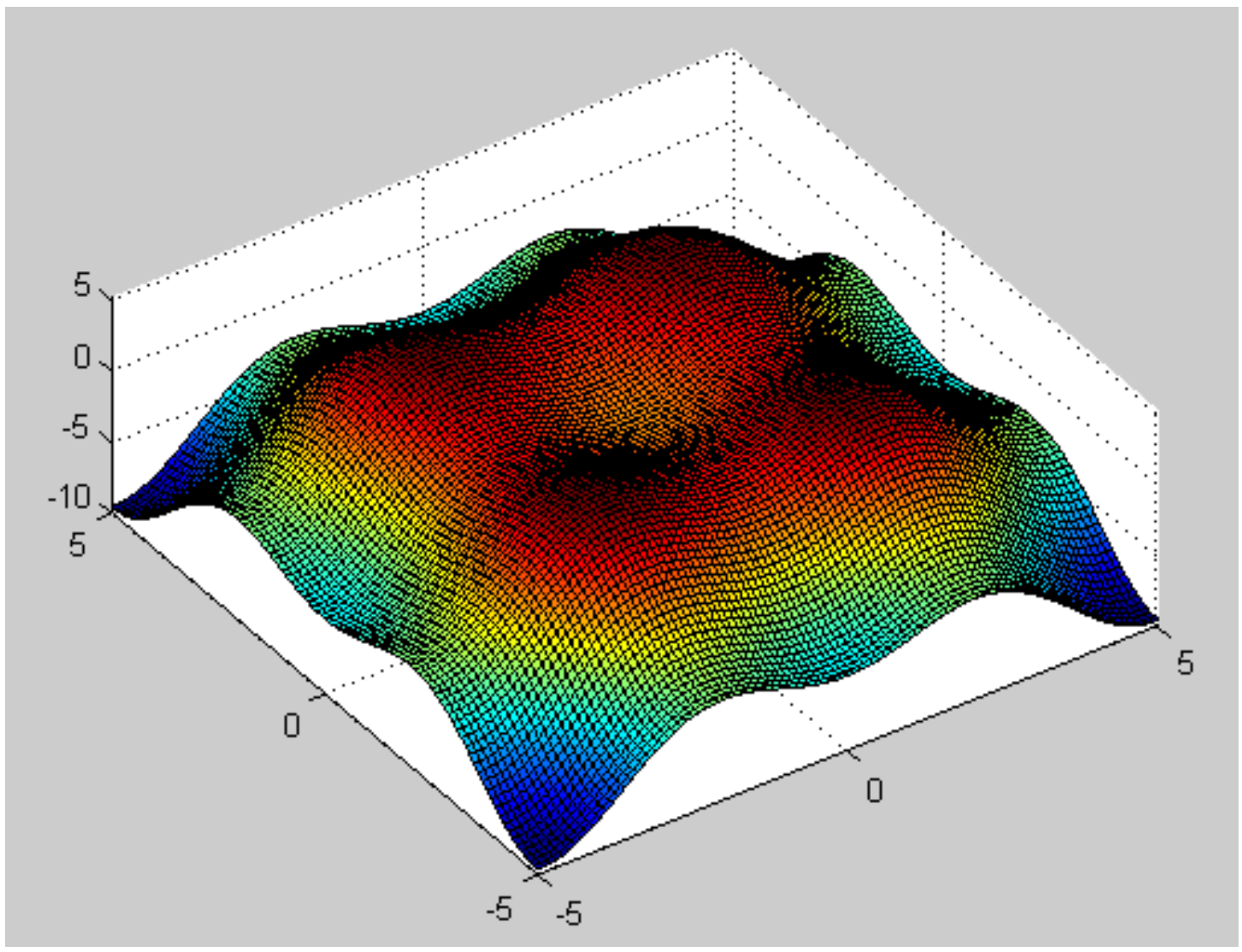
Practice 3

Maximization of the 3D function with multiple solutions.

We must search for all of the 4 maximums of the function $z = x \sin(|x|) + y \sin(|y|)$

With x = [-5; 5] and y = [-5; 5]



To find all 4 solutions I'll try to use

1. Chromosomes with 22 binary genes
   First 11 for X
   And last 11 for Y
2. Population with 20 chromosomes
3. I'll try to calculate shared fitness function with Manhattan distance d_{ij}
4. Trunked selection ½

5. Without mutations
6. Repeat until maximum shared fitness of last 10 population are equals

Code

Cromosom.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Chromosom {
    private List<Boolean> genes = new ArrayList<Boolean>();

    private static Random random = new Random();

    private Double sharedFitness;

    public Double getSharedFitness() {
        return sharedFitness;
    }

    public void setSharedFitness(Double sharedFitness) {
        this.sharedFitness = sharedFitness;
    }

    public Chromosom(){
        for(int i = 0; i < 22; i++){
            genes.add(random.nextBoolean());
        }
    }


    public Chromosom(List<Boolean> genes){
        this.genes = genes;
    }

    public static List<Chromosom> crossover(Chromosom papa, Chromosom mama){
        List<Chromosom> children = new ArrayList<Chromosom>();

        List<Boolean> firstChildGenes = new ArrayList<>();
        List<Boolean> secondChildGenes = new ArrayList<>();

        for(int i = 0; i < 22; i++){
            if(random.nextBoolean()){
                firstChildGenes.add(papa.getGenes().get(i));
                secondChildGenes.add(mama.getGenes().get(i));
            }else{
                firstChildGenes.add(mama.getGenes().get(i));
                secondChildGenes.add(papa.getGenes().get(i));
            }
        }

        children.add(new Chromosom(firstChildGenes));
        children.add(new Chromosom(secondChildGenes));

        return children;
    }
```

```
public List<Boolean> getGenes() {
    return genes;
}

public void setGenes(List<Boolean> genes) {
    this.genes = genes;
}

public double getFitness(){
    double x = 0;
    double y = 0;


    for(int i = 1; i < 11; i++){
        x += genes.get(i) ? Math.pow(2.0, i - 1)/1023.0*5.0 : 0;
        y += genes.get(i + 11) ? Math.pow(2.0, i - 1)/1023.0*5.0 : 0;
    }

    x *= genes.get(0) ? -1.0 : 1.0;
    y *= genes.get(11) ? -1.0 : 1.0;

    return x * Math.sin(Math.abs(x)) + y * Math.sin(Math.abs(y));
}

public double getX(){
    double x = 0;

    for(int i = 1; i < 11; i++){
        x += genes.get(i) ? Math.pow(2.0, i - 1)/1023.0*5.0 : 0;
    }

    x *= genes.get(0) ? -1.0 : 1.0;

    return x;
}

public double getY(){
    double y = 0;

    for(int i = 12; i < 22; i++){
        y += genes.get(i) ? Math.pow(2.0, i - 1)/1023.0*5.0 : 0;
    }

        y *= genes.get(0) ? -1.0 : 1.0;

        return y;
    }
}
```

Population.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

public class Population {

    private List<Chromosom> chromosoms = new ArrayList<Chromosom>();

    private double sigma = 9.0;

    private static Random random = new Random();
```

```
public Population() {
    for (int i = 0; i < 20; i++) {
        chromosoms.add(new Chromosom());
    }
}

public Population(List<Chromosom> chromosoms) {
    this.chromosoms = chromosoms;
}

public List<Chromosom> getChromosoms() {
    return chromosoms;
}

public void setChromosoms(List<Chromosom> chromosoms) {
    this.chromosoms = chromosoms;
}

public Population nextGeneration() {
    List<Chromosom> nextGen = new ArrayList<>();
        Collections.sort(chromosoms, (c1, c2) ->
c1.getSharedFitness().compareTo(c2.getSharedFitness()));

        for (int i = 0; i < 10; i++) {
            nextGen.addAll(Chromosom.crossover(chromosoms.get(random.nextInt(10)),
chromosoms.get(random.nextInt(10))));
        }

        return new Population(nextGen);
    }

    public double getSharedFitness(int index) {
        double result = 0;

        for (int i = 0; i < 20; i++) {
            double x = Math.abs(getChromosoms().get(index).getX() -
getChromosoms().get(i).getX());
            double y = Math.abs(getChromosoms().get(index).getY() -
getChromosoms().get(i).getY());

            double d = x + y;
            result += s(d);
        }

        return getChromosoms().get(index).getFitness() / result;
}

private double s(double d) {
    return d < sigma ? 1 - (d / sigma) : 0;
}

public void calcSharedFitnessList() {
    for (int i = 0; i < 20; i++) {
        getChromosoms().get(i).setSharedFitness(getSharedFitness(i));
    }
}

public double getMax() {
    List<Double> fitnessList = new ArrayList<>();

    for (int i = 0; i < 20; i++) {
```

```
            fitnessList.add(chromosoms.get(i).getSharedFitness());
        }

        return Collections.max(fitnessList);
    }

    public double getAverage() {
        double sum = 0;
        for (int i = 0; i < 20; i++) {
            sum += chromosoms.get(i).getSharedFitness();
        }

        return sum/20.0;
    }
}
```

plotGraph.m (Matlab code for graphic plot)

```
x = (-5 : 0.1 : 5);
y = (-5 : 0.1 : 5);
z = zeros(101, 101);

for i = 1:101
   for j = 1 : 101
       z(i, j) = x(i) * sin(x(i)) + y(j) * sin(y(j));
   end
end

surf(x, y, z);
```