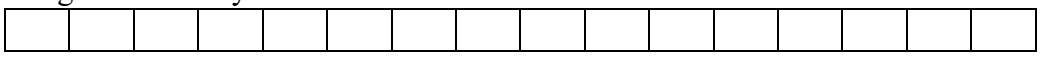
Description of Question 2

1. Basic variable:

   Init_population:1000
   Max iteration times:5000
   Mutation rate:0.2
   Crossover rate:0.5

2. encode:

   using 16-bit binary code

   The first bit stands for the positive or the negative.
   The second to the fifth bits stand for the integer part of the value, using binary representation, just meaning an integer of zero to fifteen.
   And the left bits stand for the fractional part of the value, using binary representation. Since eleven bits of binary code can represent 2047 at max, I just multiply the fraction part by 10000 to get a four-digit integer, and if it is bigger than 2047, then I divide it by 10 and make a mark so that I can decode precisely.
   For example, 5.1051 is coded as 0 0101 10000011011, -0.5050 is coded as 1 0000 00111111001 with a special mark.

3. fitness function:

   define f(x) = x*sin(x)
   for each number, its fitness is defined as 1/(f(x)+15)^3.
   Since the f(x) is definitely bigger than 15, so f(x) plus 15 is a positive number for sure. And I use cubic method to optimize it, so that the code is more efficient. Countdown is to make the smaller function value larger fitness.

4. selection method:

   For each generation, I use roulette method to select next generation.

5. crossover method:

   First, randomly generate a floating point number between 0 and 1.
   Then, if the floating point number is less than crossover_rate, randomly select two individual form the left population, use partial-mapped crossover operator to generate two offspring, and replace parent chromosome with its offspring.

6. mutation method:

   For each individual in the population:
   First, uniformly generate a floating point number between 0 and 1.
   If the floating point is less than mutation_rate, then the individual produces mutation.
   Randomly select the mutation position, and if the binary number corresponding to

this position is 0, then turn it to 1;else turn it to 0.

To make the number fit the range of [-1,15], I use a special position selection method: for a positive number, the mutation position locates on the second to the last bits; for a negative number, the mutation position locates on the sixth to the last bits.

7. end criterion:

   If the number of iterations is larger than max iteration times or the average fitness score is close to the best fitness score (Specifically if the difference of the best fitness score and the average fitness score is less than 0.01), terminate.

8. result

   After testing for many times, the value of the best fitness score is close to 11.0857(difference is less than 0.001).