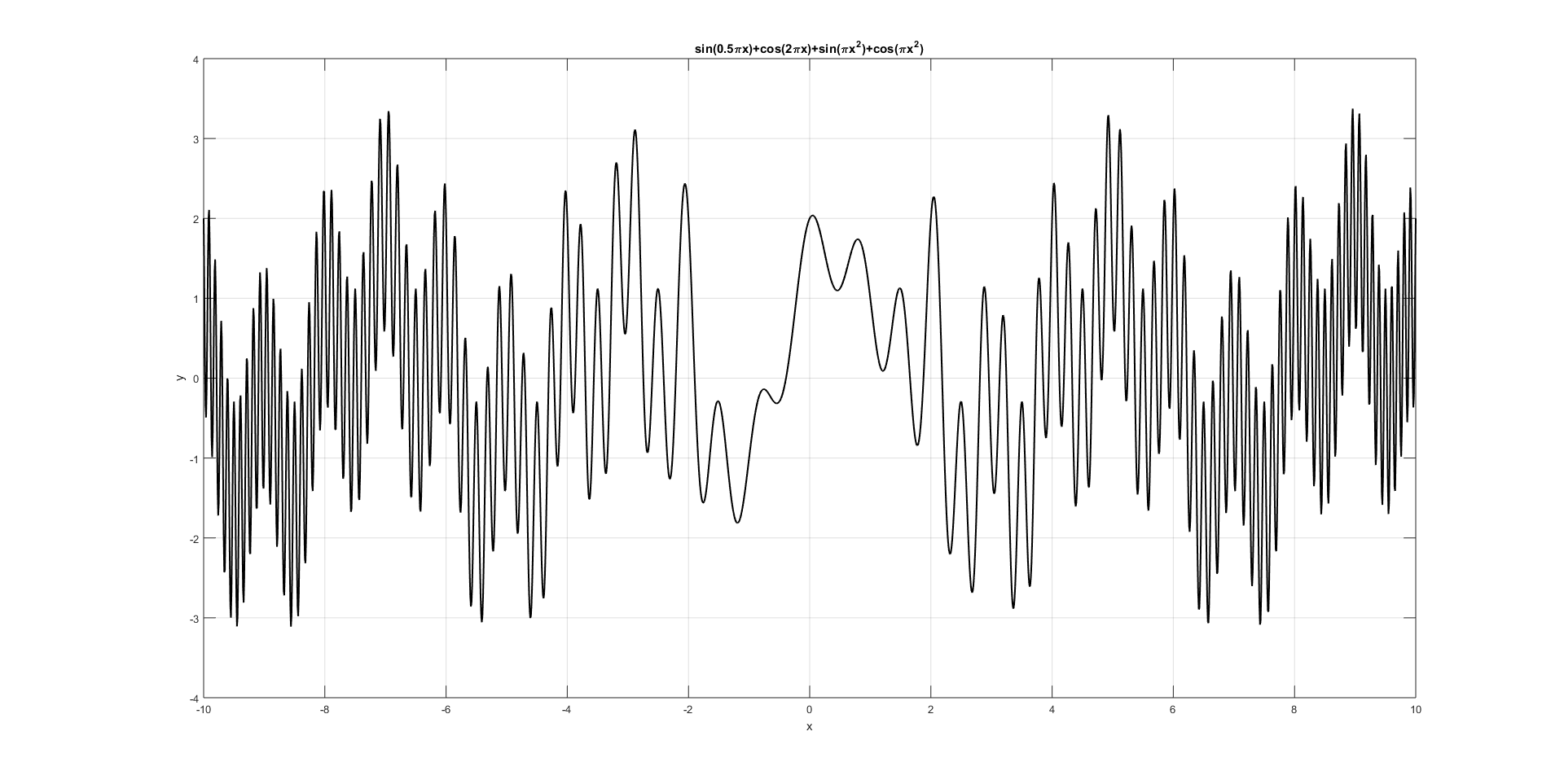
Project 1:

Find the minimized value of the function:



**Gradient method**:

Code:

learning\_rate\_Array = [0.001, 0.01, 0.1, 1];

x0\_Array = 10\*(2\*rand(1,100)-1).\*rand(1,100);

iterations = 100;

for i\_x0 = 1:numel(x0\_Array)

for i\_alpha = 1:numel(learning\_rate\_Array)

learning\_rate = learning\_rate\_Array(i\_alpha);

x(1)=x0\_Array(i\_x0);

epsilon = 0.0001;

for i=1:iterations

dy(i)=-2\*pi()\*x(i).\*sin(pi()\*x(i)^2)+2\*pi()\*x(i)\*cos(pi()\*x(i)^2)...

-2\*pi()\*sin(2\*pi()\*x(i))+(pi()/2)\*cos((pi()\*x(i))/2);

if abs(dy(i)) <= epsilon

y(i)=sin(pi()\*x(i)\*0.5)+cos(2\*pi()\*x(i))+sin(pi()\*x(i)^2)+cos(pi()\*x(i)^2);

[n,xx,yy,dyy] = deal(i,x(i),y(i),dy(i));

disp(['Results for x0 = ', num2str(x(1)), ', alpha = ', num2str(learning\_rate) ...

': n = ', num2str(n), ', ', 'xx = ', num2str(xx), ...

', ', 'yy = ', num2str(yy), ', ', 'dyy = ', num2str(dyy)])

break

else

x(i+1)=x(i)-learning\_rate\*dy(i);

y(i)=sin(pi()\*x(i)\*0.5)+cos(2\*pi()\*x(i))+sin(pi()\*x(i)^2)+cos(pi()\*x(i)^2);

end

if i == iterations

disp(['Results for x0 = ', num2str(x(1)), ', alpha = ', num2str(learning\_rate) ...

': no results found in ', num2str(iterations), ' iterations'])

end

end

end

end

The lowest y value was calculated by gradient method for **x = -4.6072**; f(-4.6072)= -3.0075 in 11 iterations for learning rate LR=0.001 and starting point x0=-4.6972

Enlarging number of maximum iterations did not yield better results, the algorithm gets stuck at local minima very often.

Minimized value (gradient method):

**Genetic Algorithm:**

Fitting function:

function PI = GA\_fitfunP1(chro)

global MIN\_offset

MIN\_offset = 10;

x = chro;

z = sin(pi()\*x\*0.5)+cos(2\*pi()\*x)+sin(pi()\*x^2)+cos(pi()\*x^2);

PI = MIN\_offset - z;

And the code to execute all the function is as following:

popu\_size=70;

bit\_length=40;

gene\_no=1;

range=[-10;10];

fitfcn='GA\_fitfunP1';

generation\_no=100;

crossover\_rate=0.7;

mutate\_rate=0.02;

elite=1;

[popu, popu\_real, fitness, upper, average, lower, BEST\_popu]...

=GA\_genetic(popu\_size, bit\_length, gene\_no, range, fitfcn, ...

generation\_no, crossover\_rate, mutate\_rate, elite);

global MIN\_offset

minfitness=MIN\_offset-upper;

[minimum\_f,generation]=min(minfitness)

minimum\_x=BEST\_popu(generation)

t=1:generation\_no;

plot(t,minfitness,'\*:')

title('Minimum of f(x)=sin(0.5\pix)+cos(2\pix)+sin(\pix^2)+cos(\pix^2)');

xlabel('Generation')

ylabel('f(x)')

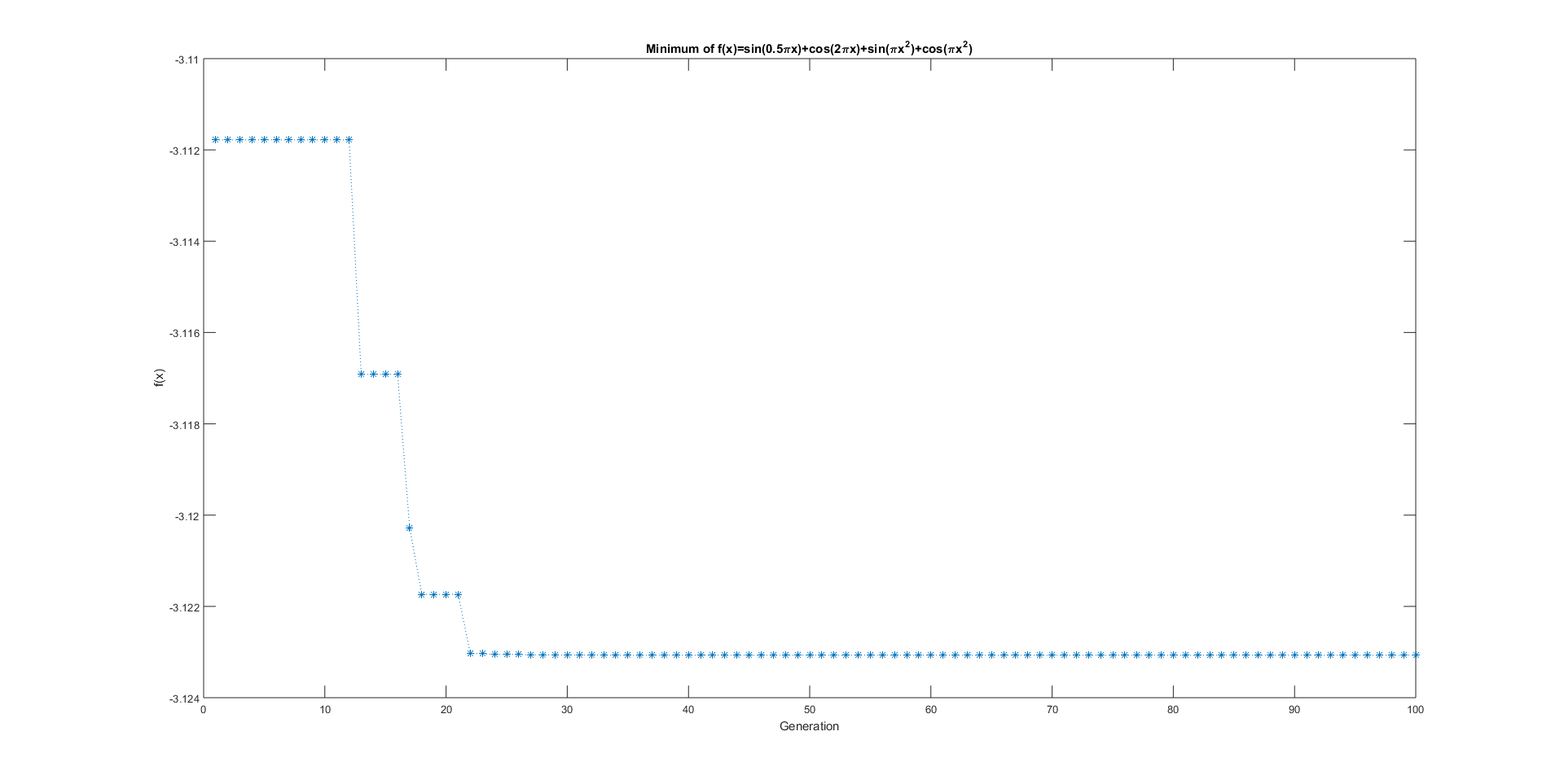
Results:

>> EX1

minimum\_f = -3.1231

generation = 33

minimum\_x = -9.4474



So global minimum was achieved in 33rd generation (20-32 minimized value decreases slowly):

**Gradient Method and Genetic Algorithm comparison:**

1. Accuracy: for the function given, genetic algorithm performs better than gradient method and returns smaller minimized value.
2. Gradient method for the function given usually gets stuck at local minima. Genetic algorithm tends to get out of the local minima because of mutation.
3. Initialization of gradient method is crucial for the performance of the method. For learning rates larger than 0.01 gradient method does not converge at all in given example. It means that algorithms always oversteps local minima in this case. Usually, the minimized value was close to the starting point of the algorithm. So, initial parameters influenced returned minimal value heavily. Although genetic algorithm also requires choosing initial parameters, these parameters influence how genetic algorithm acts (e.g. whether it’s susceptible to get stuck at local minima or not, whether it acts random or not, etc.)
4. Genetic algorithm is a heuristic method which means that it provides the best fitted value in specified number of iterations. On the other hand, gradient descent is not guaranteed to converge at all depending on the initialization parameters.