clc

clear

n = 2^30-1;

x = 0:1000:n;

plot(x/n, (x/n).^2, 'r', x/n, (x/n).^10, 'b--', 'LineWidth', 2)

xlabel('Normalized x (x/c)');

ylabel('Normalized Fitness F(x)');

title('Fitness Function to Maximize');

legend('x^2', 'x^10');

%% Genetic Algorithm

max\_generation = 25;

length\_chromosome = 30;

population\_size = 30;

pcross = 0.6;

pmutate = 0.0333;

c = 2^length\_chromosome - 1;

Fx = (x/c).^10;

% determine encoding, how many bits

% num\_bits = log(x(length(x)))/log(2); % = 30, hardcoded as length\_chromosomes

% set initial population & evaluate fitness of initial population

% I'd have to loop again if I separated these steps

% Assumes chromosomes are encoded with LSB first

pop = zeros(population\_size, length\_chromosome);

x\_decode = zeros(1, population\_size);

fitness = zeros(1, population\_size);

for ii=1:population\_size,

powerof2 = 1;

for jj=1:length\_chromosome,

pop(ii,jj) = (rand <= 0.5); % generate chromosome bit

if (pop(ii,jj)),

x\_decode(ii) = x\_decode(ii) + powerof2; % decode chromosome bit

end;

powerof2 = 2\*powerof2;

end;

% calculate fitness of decoded chromosome (run through Fx)

fitness(ii) = (x\_decode(ii)/c).^10;

end;

sum\_fitness = sum(fitness);

average\_fitness = sum\_fitness/population\_size;

max\_fitness = max(fitness);

min\_fitness = min(fitness);

% loop

for kk=1:max\_generation

% select mating pool M(i) from population for offspring reproduction

num\_parents = population\_size;

parents = zeros(1, num\_parents);

for ii=1:num\_parents,

% calculate random variable

random\_variable = rand \* sum\_fitness;

jj = 0;

partial\_sum = 0;

while (partial\_sum < random\_variable && jj < length(fitness)),

jj = jj + 1;

partial\_sum = partial\_sum + fitness(jj);

end;

if (partial\_sum >= random\_variable), % b/c it's possible to exit loop above without this happening

parents(ii) = jj;

else

ii = ii - 1; % if no chromosome selected then try again, final # should = num\_parents we're trying for?

end;

end;

child = pop;

for ii=1:length(parents)-1,

% mutate or crossover first ????

% M(i) mutate - perturb the mated population stochastically

for bb=1:length\_chromosome

if (rand <= pmutate)

child(parents(ii),bb) = ~pop(parents(ii),bb);

end

if (rand <= pmutate)

child(parents(ii+1),bb) = ~pop(parents(ii+1),bb);

end

end

% M(i) crossover - recombine genes of selected parents

if (rand <= pcross)

cross\_point = randi([1 length\_chromosome]);

for jj=cross\_point:length\_chromosome

child(parents(ii),jj) = pop(parents(ii+1),jj);

child(parents(ii+1),jj) = pop(parents(ii),jj);

end

end

ii = ii + 1; % create children in pairs

end;

% update population

pop = child;

% evaluate new fitness

x\_decode = zeros(1, population\_size);

for ii=1:population\_size,

powerof2 = 1;

for jj=1:length\_chromosome,

if (pop(ii,jj)),

x\_decode(ii) = x\_decode(ii) + powerof2; % decode chromosome bit

end;

powerof2 = 2\*powerof2;

end;

% calculate fitness of decoded chromosome (run through Fx)

fitness(ii) = (x\_decode(ii)/c).^10;

end;

sum\_fitness = sum(fitness);

max\_fitness = [max\_fitness max(fitness)];

average\_fitness = [average\_fitness sum\_fitness/population\_size];

min\_fitness = [min\_fitness min(fitness)];

end

%% Plot stuff

figure()

hold on

plot(max\_fitness)

plot(min\_fitness)

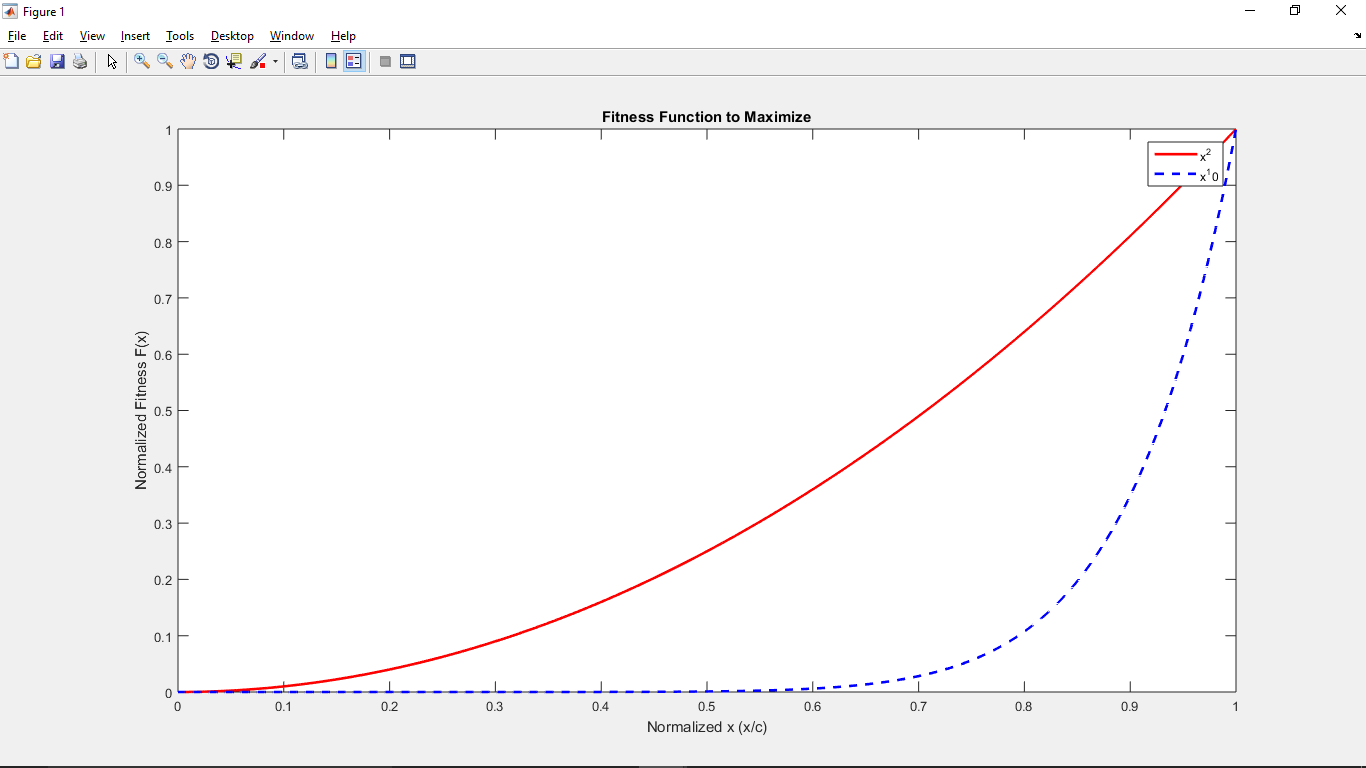
plot(average\_fitness)

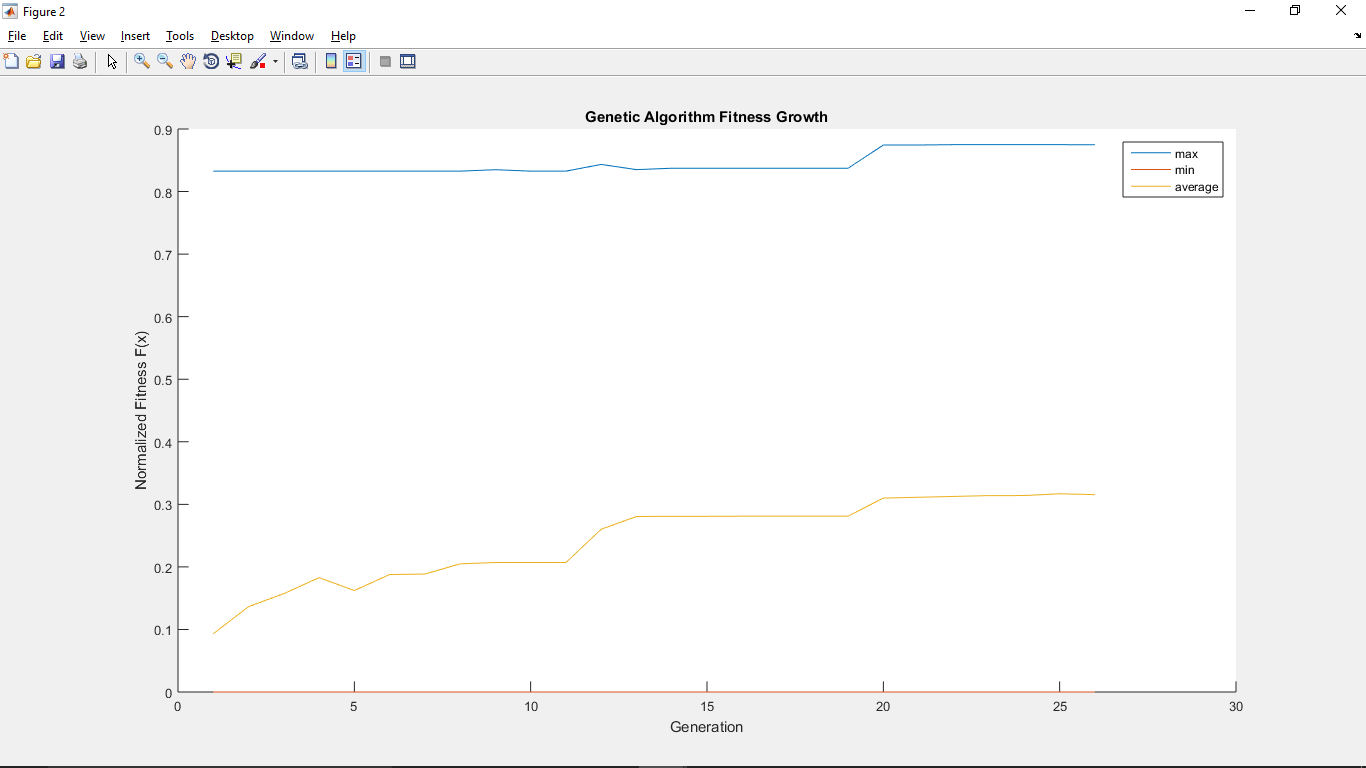
xlabel('Generation');

ylabel('Normalized Fitness F(x)');

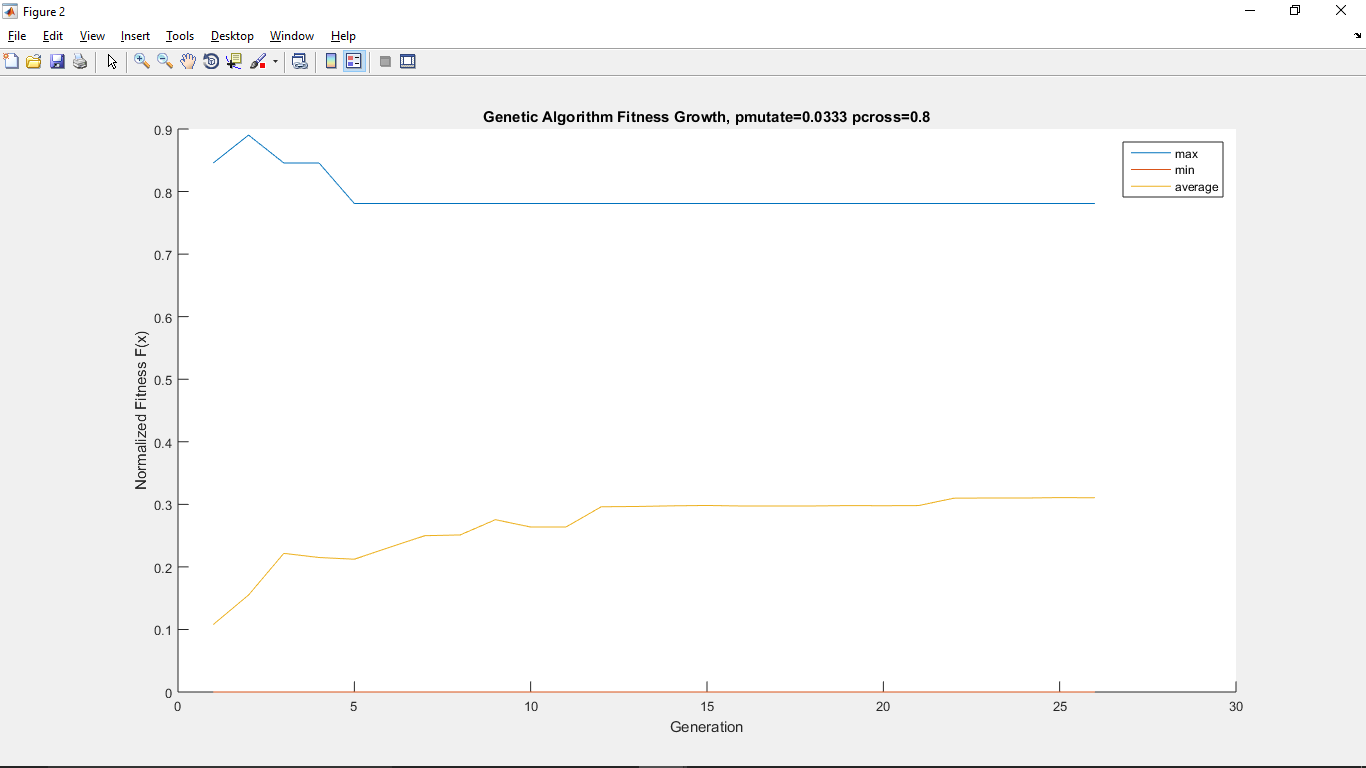
title('Genetic Algorithm Fitness Growth, population 40');

legend('max', 'min', 'average');

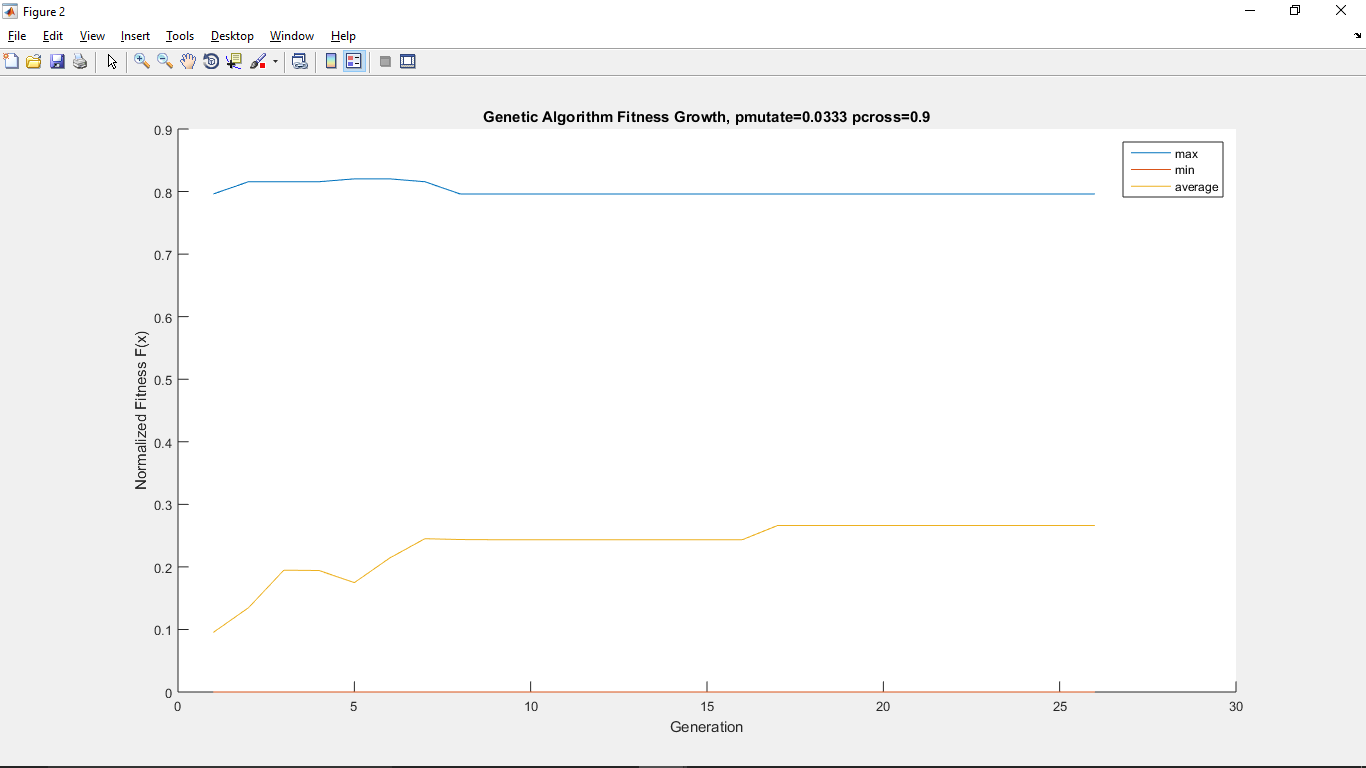




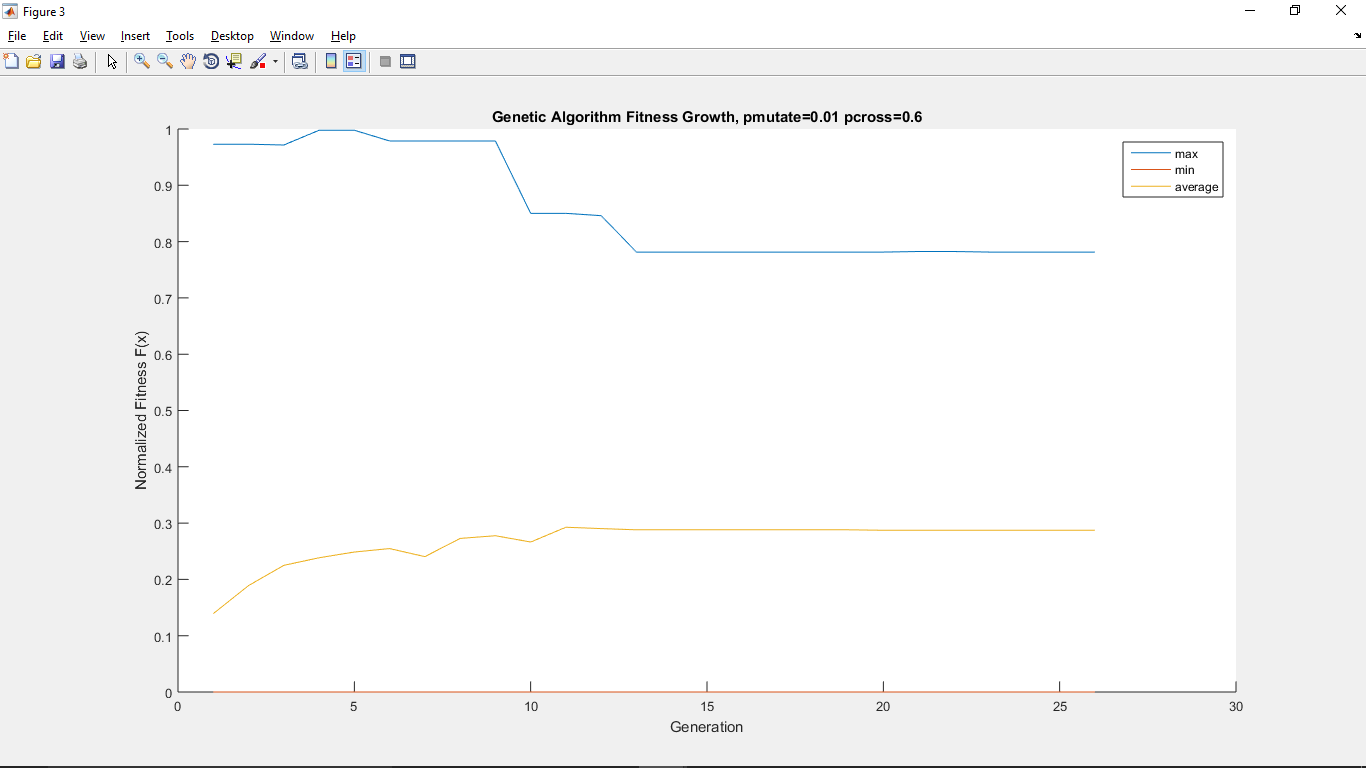
pcross = 0.8, pmutate=0.0333



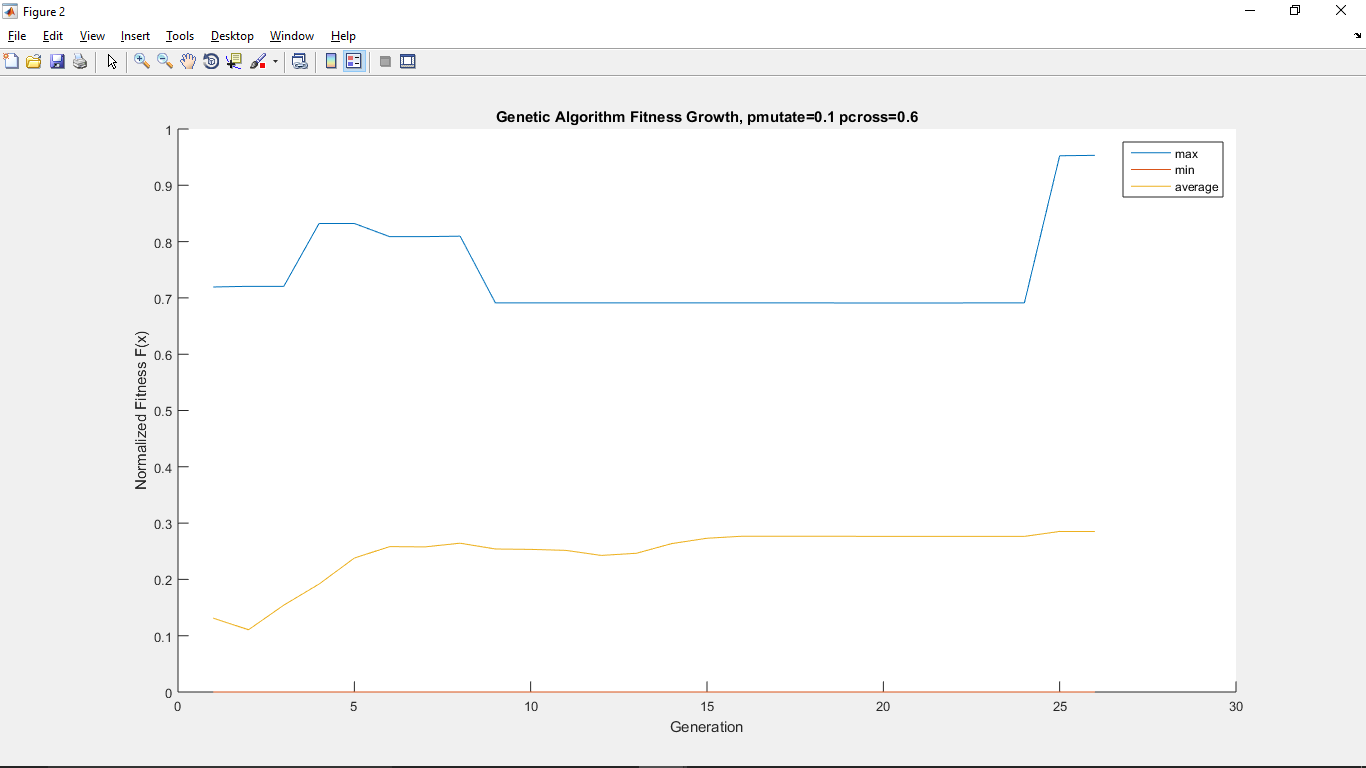
pcross = 0.9, pmutate=0.0333



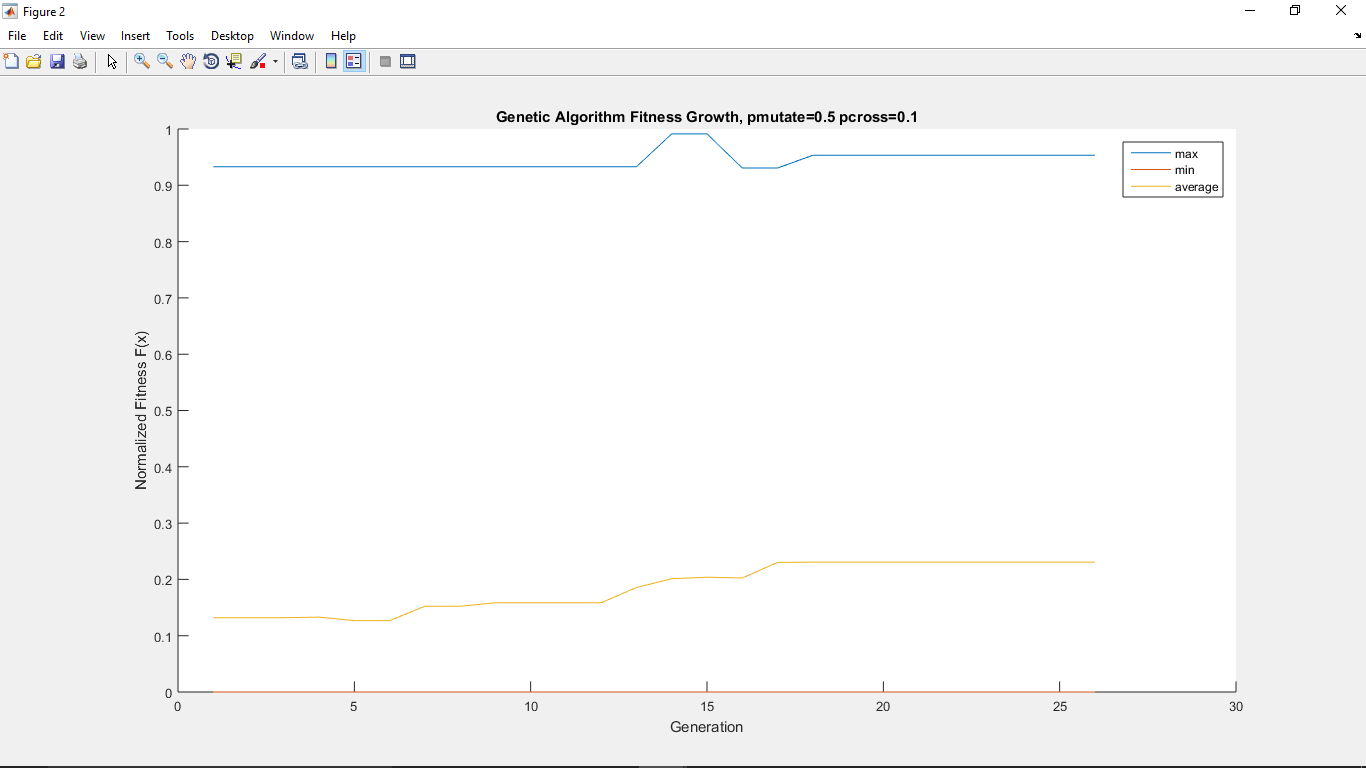
pcross = 0.6, pmutate=0.01



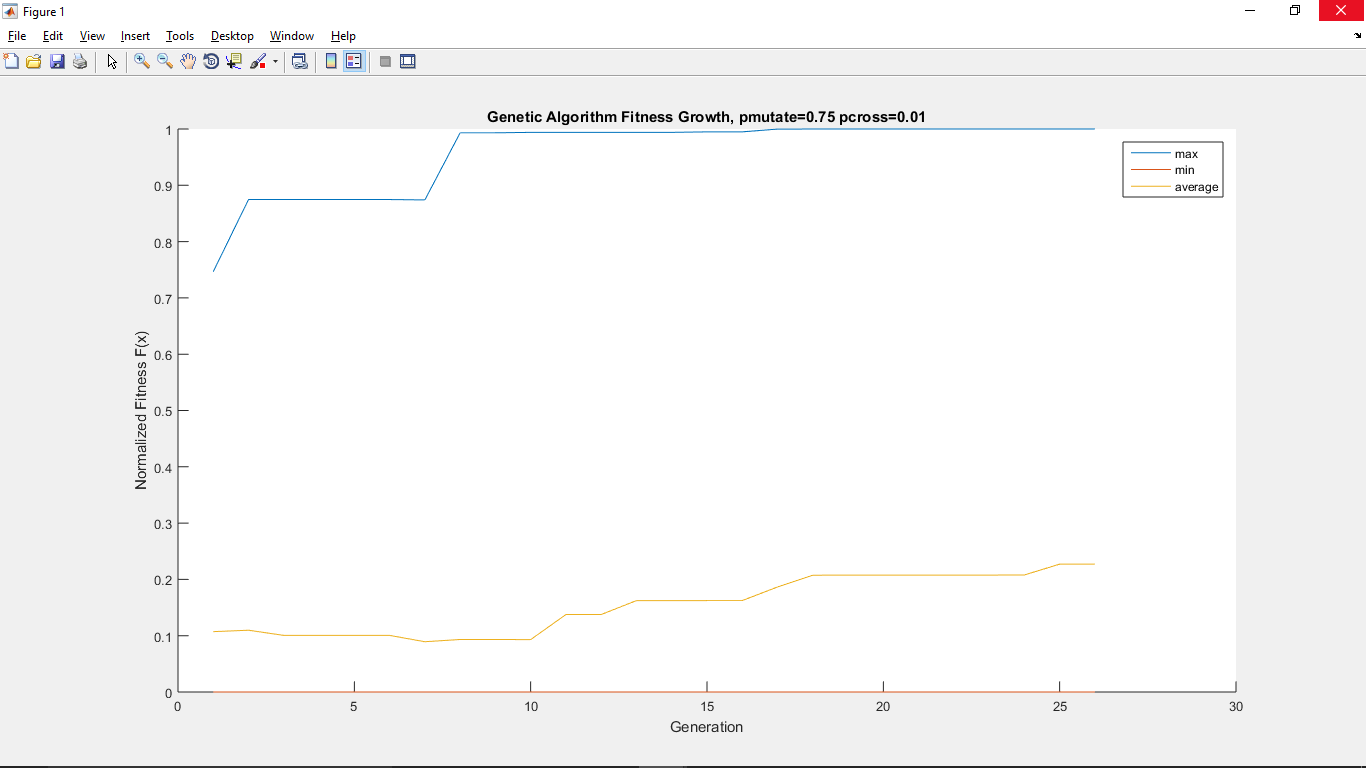
pcross = 0.6, pmutate=0.1



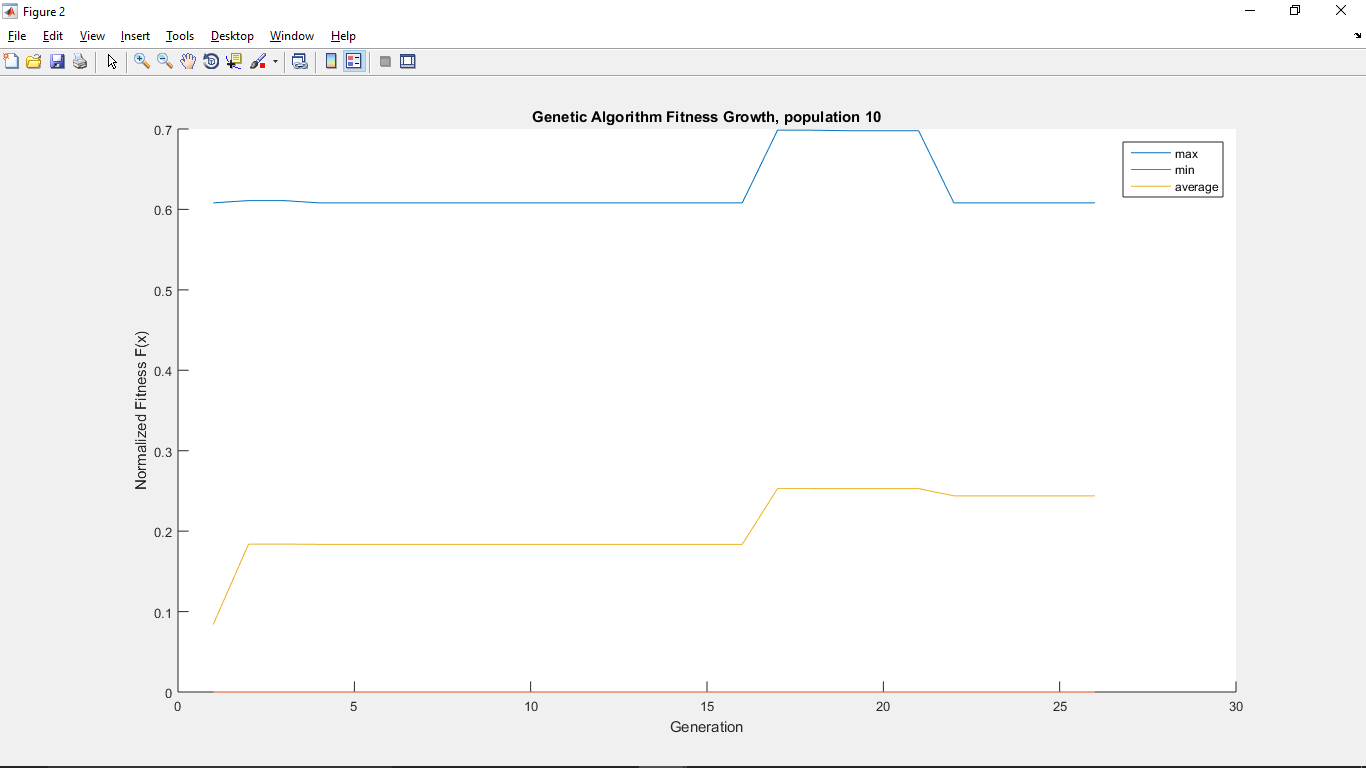
pcross = 0.1, pmutate=0.5



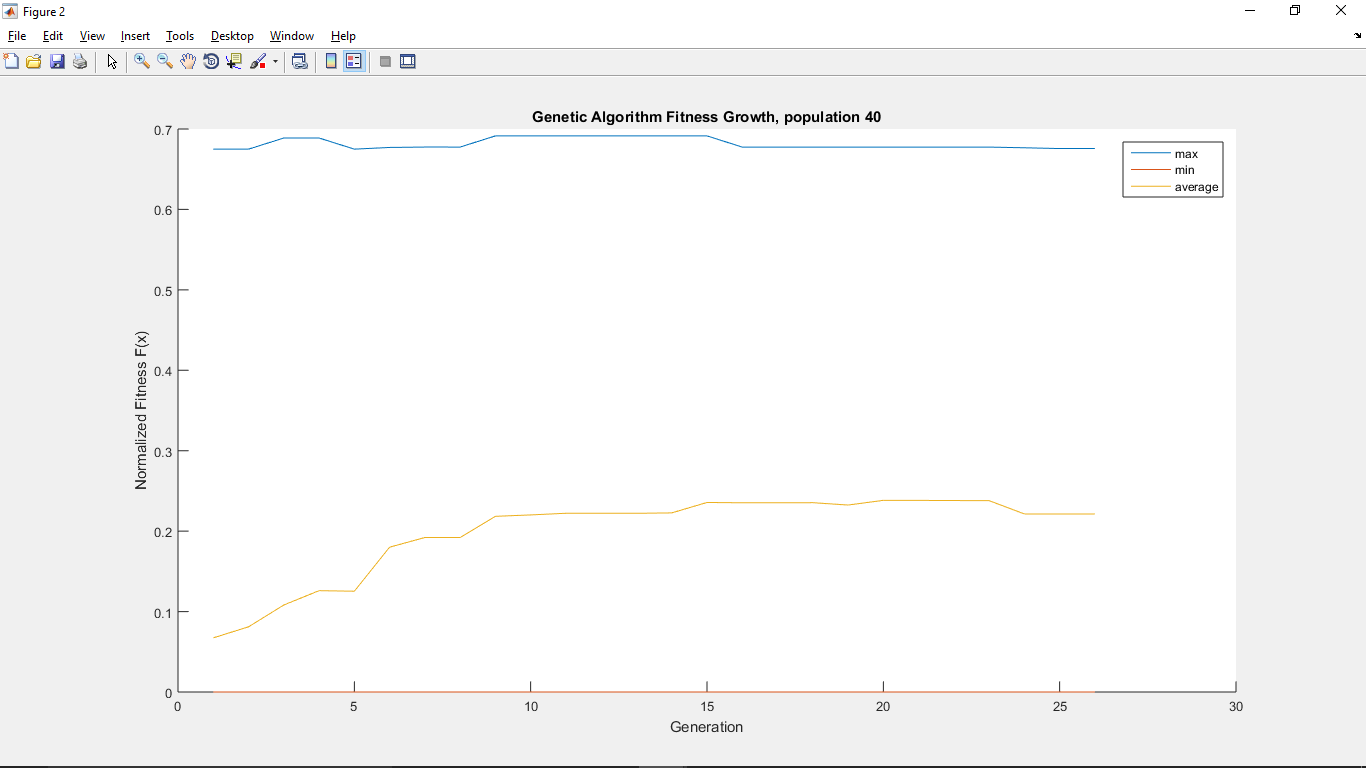
pcross = 0.01, pmutate=0.75



Population 10



Population 40



Results don’t seem to change much depending on pcross, pmutate, but are very random between runs. Clearly something not quite right in my code…