CSE 848 Home Assignment 1

Submitted by: Ritam Guha (MSU ID: guharita)

Date: January 26, 2021

1. We are interested in finding the minimum solution of the following five-variable (n = 5) Rastrigin function starting from 100 random initial points in $x_i \epsilon [-5.12, 5.12]$:

$$f(x) = 10n + \sum_{x=1}^{n} x_i^2 - 10\cos(\alpha \pi x_i)$$

Use $\alpha = 0.25$, 1.0, and 2.0. Apply *fminsearch()* to solve each instance with maximum function evaluations of 5,000 and maximum iterations of 2,000 for each run. For each α, report the best solution of 100 runs. Can you explain the results obtained?

Solution: The given Rastrigin function constructs a 5-dimensional ($n = 5$) multivariable, unconstrained minimization problem. So, Nelder-Mead's simplex search algorithm can be used to solve this problem. The algorithm has an implementation under the name *fminsearch()* in MATLAB. As suggested in the question, the maximum number of function evaluations and maximum number of allowable iterations have been set to 5000 and 2000 respectively. following code snippet has been used to solve the given optimization problem.

```
% Testing conditions
max_iter = 2000;
max_func_eval = 5000;
num_runs = 100;
n = 5;  % dimension of the objective function f(x)
a = -5.12;  % lb on x
b = 5.12;   % ub on x

% fminsearch options
options = optimset('MaxIter', max_iter, 'MaxFunEvals',
    max_func_eval);

min_fval = Inf; % intiialize min value of f
min_x = Inf;    % value of x corresponding min f

```

```
for run_no = 1:num_runs
    % Create random initial point in [a, b]
    x0 = a + (rand(1, n) * (b-a));
    [x, fval] = fminsearch(@f, x0, options);     % use
        Nelder-Meads search algorithm

    if(fval < min_fval)
        min_fval = fval;
        min_x = x;
    end
end

disp(['Min f(x): ', num2str(min_fval)]);
disp(['x: ', num2str(min_x)]);


function [func_val] = f(x)
    % Formulation of n-dimensional Rastrigin function
    alpha = 2.0;   % set the value of alpha
    n = size(x, 2);
    func_val = 10*n + sum(x.^2 - 10*cos(alpha * pi * x));
end
```

After running *fminsearch()* with 100 different initializations for different values of α, the best solutions have been tabulated in Table 1. In order to demonstrate the convergence prowess of *fminsearch()*, the reduction in the functional value over the iterations for a single run with α=0.25, 1.0 and 2.0 are displayed in Figure 1.

α	**Min f(x)**	**x**
0.25	4.05E-09	-1.6056e-05, 2.0453e-05, -1.1009e-06, 1.2019e-05, -1.3062e-05
1.0	7.8409	-2.9483e-05, 2.8664e-05, 1.9602, 8.6819e-07, -1.9602
2.0	13.9294	3.5673e-05, -4.1348e-05, -0.99493, 2.9849, -1.9899

Table 1: The min f(x) and corresponding x obtained after 100 runs of *fminsearch()* with different values of α

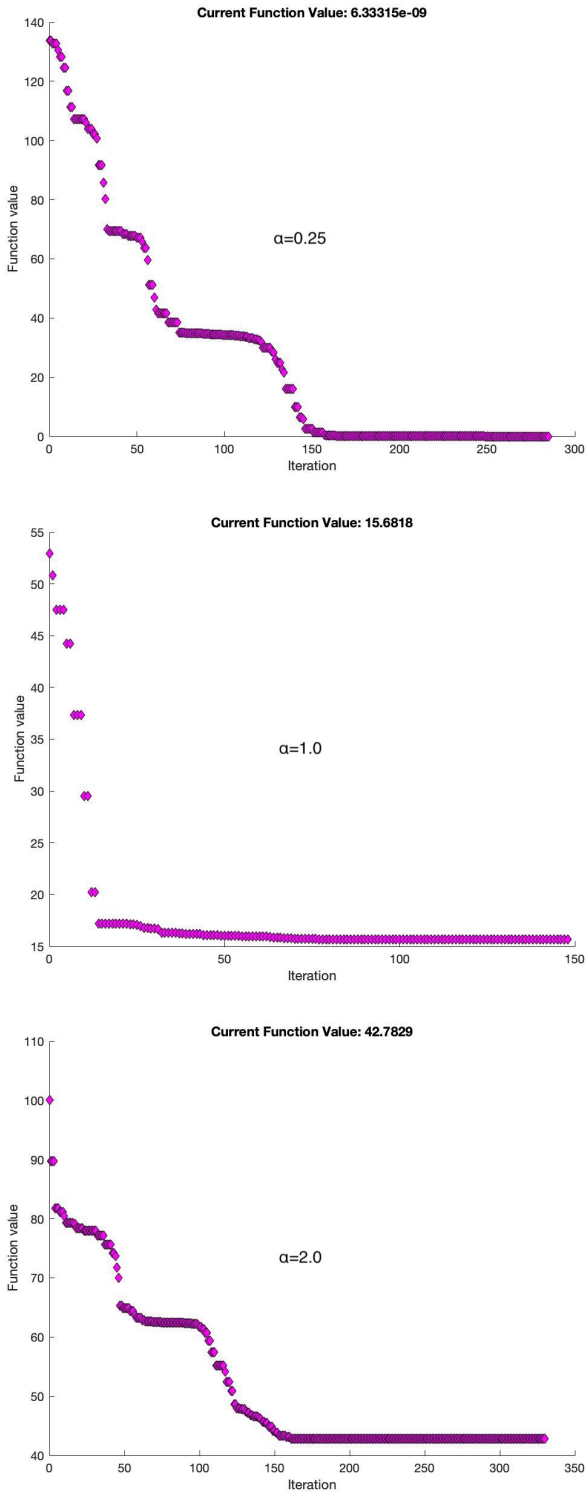
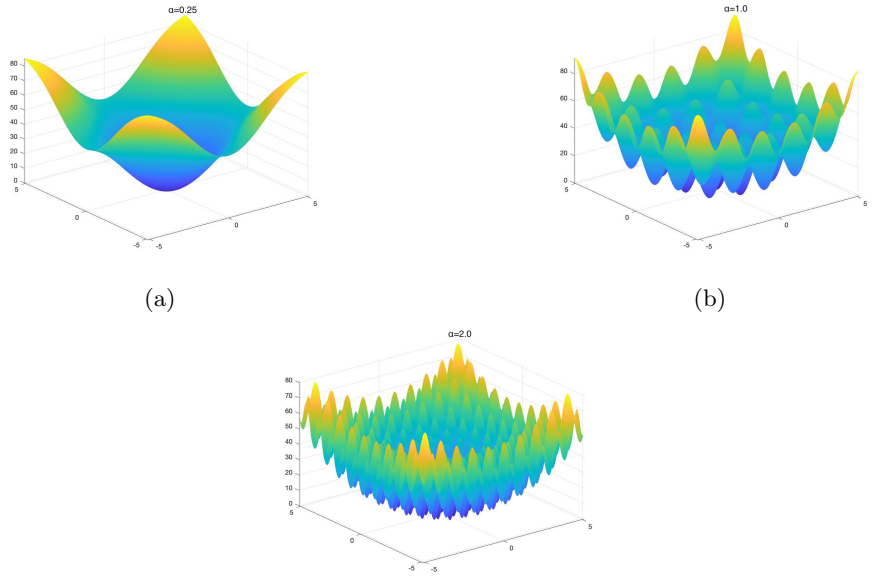


Figure 1: Reduction in function values over the iterations for α=0.25, 1.0 and 2.0 respectively.

From the results obtained using the *fminsearch()* method, we can see that the algorithm works the best when α=0.25 and it performs worse as α value increases. In order to investigate the validity of such results, I plotted the 2-dimensional Rastrigin function for different values of α. These plots are presented in Figure 2.



(a) (b)

(c)

Figure 2: Rastrigin function plotting for α=0.25 (a), 1.0 (b) and 2.0 (c) respectively.

From the plots of these functions, it is clearly visible that as the α value increases, the number of local optima increases as well. So, it becomes difficult for the algorithm to reach the global optimum. The final value also depends heavily on the initial point selection. If the initial point is closer to the global optimum (or its basin), the algorithm can easily find the optimum after some iterations, but if that's not the case, it can get stuck in a local optimum. This is the reason why the quality of the final solution decreases as value of α increases.

2. For the following constrained minimization problem, find the minimum solution using *fmincon()* routine.

$$\begin{aligned} \text{Minimize } f(x_1, x_2) &= (x_1 + 1)^2 + (x_2 + 1)^2 \\ \text{Subject to: } g_1(x_1, x_2) &= (x_1 - 1)^2 + 4x_2^2 \leq 5, \\ g_2(x_1, x_2) &= x_1 - x_2 \geq 1, \\ g_3(x_1, x_2) &= x_1 + 2x_2 \leq 2. \end{aligned}$$

Start with an initial solution (1, 1). Plot the history of intermediate solutions and also plot the reduction of objective values with iteration.

Solution: The given problem is a multivariable, constrained, non-linear minimization problem. In order to make the problem definition suitable for *fmincon()* function, at first, the $\geq$ type constraint g_2 is modified to $\leq$ type by multiplying -1 to both sides of the constraint. Then the matrix formulation (for the linear part) of the problem becomes:

$$\text{A} = \begin{bmatrix} -1 & 1 \\ 1 & 2 \end{bmatrix}, \text{B} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

For the non-linear constraints, a function is designed to return the value of c (for non-equality) and *ceq* (for equality) whenever called. This completes the process of formulation and then *fmincon()* is called using all the parameters. Starting with the point $(1, 1)$, a minimum function value of 1 is obtained at the point $x = (2e - 06, -1)$. The intermediate history plot and function reduction plot are displayed in Figure 3 and Figure 4 respectively.

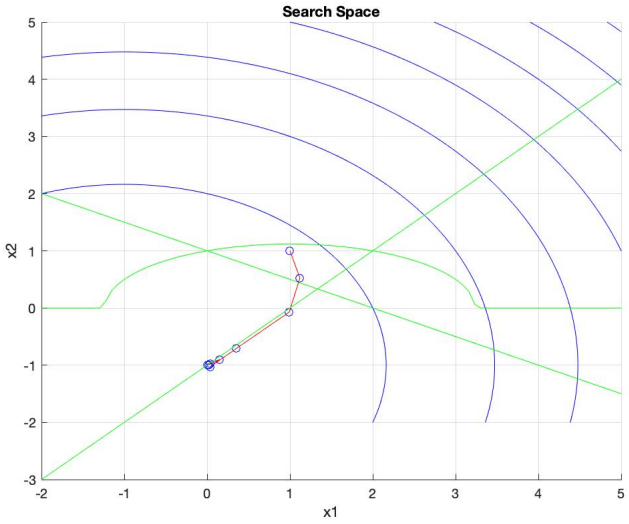


Figure 3: History of intermediate solutions for x

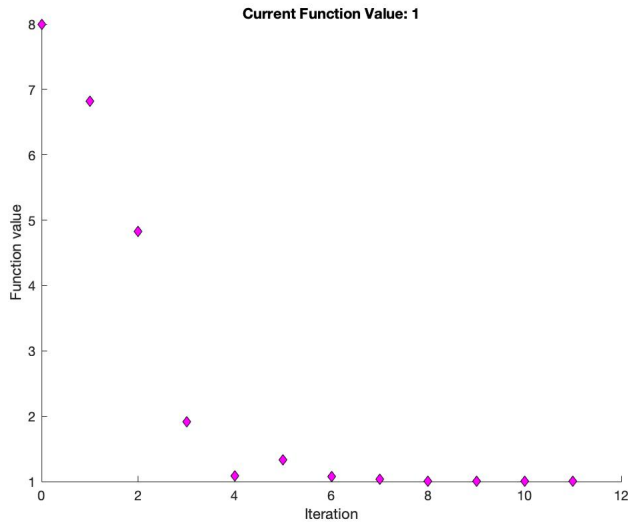


Figure 4: Reduction of f(x) over iterations

I tried checking the performance of the algorithm with different initial points. For this purpose, I have used two different points (2, 2) and (-1, -3). The history of the intermediate solutions for these two initial points are presented in Figure 5 and Figure 6.

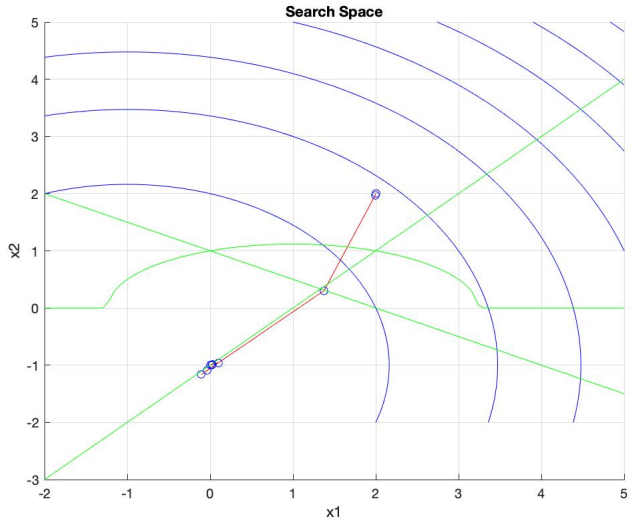


Figure 5: History of intermediate solutions for x

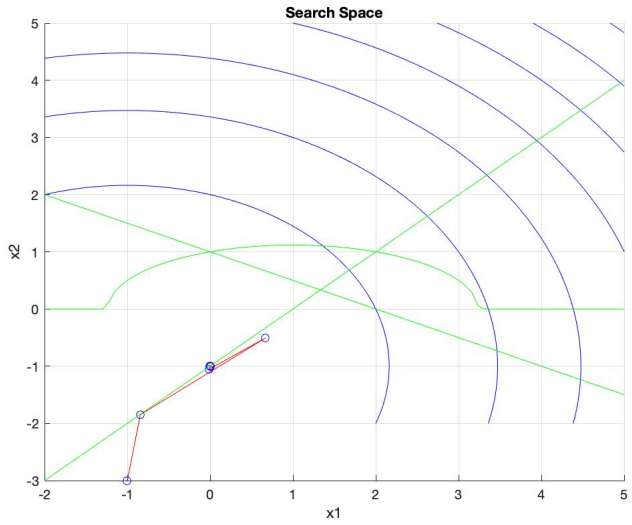


Figure 6: History of intermediate solutions for x

The code used to obtain minimized solution for the given problem is provided below:

```
x0 = [1, 1];    % initial point
[x, fval, history] = fmincon_driver(x0);

% Plot History
x1 = linspace(-2, 5);
x2 = linspace(-2, 5);
[X1, X2] = meshgrid(x1, x2);
Z = (X1+1).^2 + (X2+1).^2;

figure;
hold on;
grid on;

title('Search Space');
xlabel('x1');
ylabel('x2');

plot(history(:,1),history(:,2),'r-');
plot(history(:,1),history(:,2),'bo');
contour(X1,X2,Z,'b');
plot(x1,sqrt((5-(x1-1).^2)/4),'g-');
plot(x1,x1-1,'g-');
plot(x1,(2-x1)/2,'g-');


function [x, fval, history] = fmincon_driver(x0)

    % Objective Function
    f = @(x) ((x(1)+1).^2 + (x(2)+1).^2);

    % Constraint Formation

            % linear
    A = [-1, 1; 1, 2];
    B = [-1; 2];
    Aeq = [];
    Beq = [];
    lb = [];
    ub = [];

            % non-linear
    nonlcon = @nlconstraint;

    history = x0;
    options = optimset('Display', 'Iter', 'PlotFcns',
        @optimplotfval, 'OutputFcn', @output_check);
```

```
46      [x, fval] = fmincon(f, x0, A, B, Aeq, Beq, lb, ub,
            nonlcon, options);
47
48      function [stop] = output_check(x, optimValues, state)
49          % nesting function to store history
50          stop = false;
51          if isequal(state, 'iter')
52              history = [history; x];
53          end
54      end
55  end
56
57
58  function [c, ceq] = nlconstraint(x)
59      % function returning non-linear constraints
60      c = (x(1)-1)^2 + 4*x(2)^2 - 5;
61      ceq = [];
62  end
```

3. First, solve the following problem using *linprog()* routine of Matlab.

$$\begin{aligned} \text{Maximize } & x_1 + 4x_2, \\ \text{subject to: } & x_1 + 5x_2 \leq 10, \\ & 3x_1 + x_2 \leq 15, \\ & x_1 + 2x_2 \geq 1, \\ & x_1, x_2 \geq 0. \end{aligned}$$

Then, solve it using MATLAB's *intlinprog()* routine for which the second variable is real-valued, but the first variable can only take an integer value. Show the optimal solutions on a $x_1 - x_2$ plot of feasible region and explain the validity of the obtained optimal solutions.

Solution: The given problem is a multivariable, constrained maximization problem. In order to solve the problem using *linprog()* or *intlinprog()*, first certain modifications were made to the problem description to convert it to a minimization problem and make it suitable to feed to the algorithm implementations. For example, the third constraint is a $\geq$ type, so it was changed to a $\leq$ type by mutiplying both the sides by -1. Finally, our A, B and C matrices become:

$$\text{A} = \begin{bmatrix} 1 & 5 \\ 3 & 1 \\ -1 & -2 \end{bmatrix}, \text{B} = \begin{bmatrix} 10 \\ 15 \\ -1 \end{bmatrix}, \text{C} = \begin{bmatrix} -1, -4 \end{bmatrix}$$

These matrices are fed directly to the *linprog()* routine and it obtained a maximum function value of 8.9286 with $x = (4.6429, 1.0714)$.

For the next part of the question, the first variable was fixed to integer values by setting *intcon* = [1]. Using *intlinprog()*, a maximum function value of 8.8 was achieved corresponding to the point $x = (4, 1.2)$. A summary of the results is provided in Table 2.

Procedure	**Max f(x)**	**x**
linprog	8.9286	4.6429, 1.0714
intlinprog, intcon=[1]	8.8	4, 1.2

Table 2: The max f(x) and corresponding x values obtained via *linprog()* and *intlinprog()*

.

The results were plotted in the search space in Figure 7. The green lines indicate the constraints of the problem and red lines denote the objective function. The optimal solutions of *linprog()* and *intlinprog()* are plotted in the graph using blue and red stars.

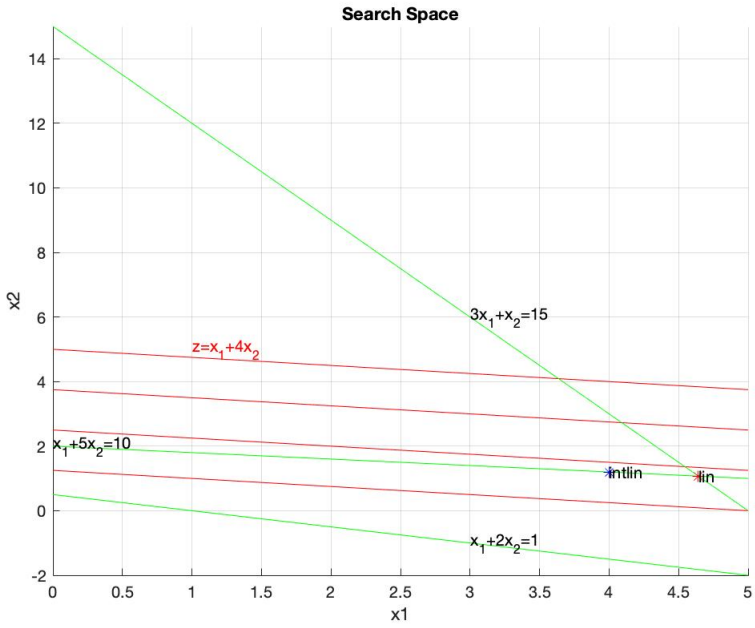


Figure 7: Solution plotting in feasible region

From the obtained results, it is clear that restricting the first variable to integer values only has reduced the overall solution quality. Restricting x_1 to integer space reduces the number of values it can take. For convex optimization problems (when functions are convex and the feasible variable space is convex), the most optimal solutions are found at the corner points of the boundary. The given problem is convex in nature and the most optimal corner point does not have integral value for x_1 (both x_1 and x_2 are real in nature for the most optimal point) That is why restricting x_1 to integer values for *intlinprog()* makes the optimal solution out of scope and hence we get a better solution for *linprog()* over *intlinprog()*.

The code used to solve the problem and plot the results is presented below:

```
% Setting up Coefficients of Obj Function and Constraints
C = [1, 4];
A = [1, 5; 3, 1; -1, -2];
B = [10, 15, -1];

% Linear Programming approach
[x_lin, fval] = linprog(-C, A, B);
x_lin = x_lin';

```

```
disp(['Max f(x): ', num2str(-fval)]);
disp(['x: ', num2str(x_lin)]);

% Restricting first variable to integer values for
    Integer Programming
intcon = [1];
[x_intlin, fval] = intlinprog(-C, intcon, A, B);
x_intlin = x_intlin';

disp(['Max f(x): ', num2str(-fval)]);
disp(['x: ', num2str(x_intlin)]);

% Plotting feasible regions
x1 = linspace(0, 5);
x2 = linspace(0, 5);
[X1, X2] = meshgrid(x1, x2);
Z = X1 + 4*X2;

figure;
hold on;
grid on;

title('Search Space')
xlabel('x1')
ylabel('x2')
contour(X1, X2, Z, 'r');
plot(x1, (10-x1)/5, 'g-');
text(0,2,'x_1+5x_2=10')
text(1,5,'z=x_1+4x_2','Color','r')
plot(x1, 15-3*x1, 'g-');
text(3,6,'3x_1+x_2=15')
plot(x1, (1-x1)/2, 'g-');
text(3,-1,'x_1+2x_2=1')
plot(x_lin(1), x_lin(2), 'r*');
text(x_lin(1), x_lin(2), 'lin');
plot(x_intlin(1), x_intlin(2), 'b*');
text(x_intlin(1), x_intlin(2), 'intlin');
hold off;
```