Solving ZDT1

Constraints & Solver

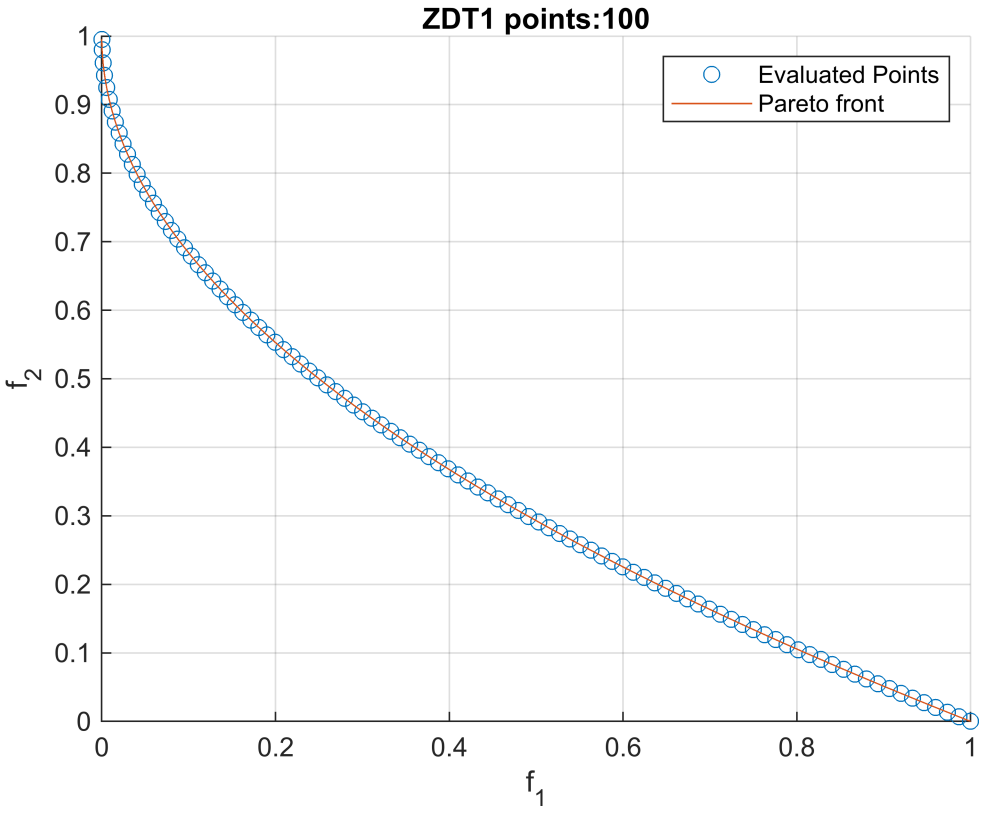
```
N = 9; % Variables in x vector
Functions = {@f1, @f2}; % Objective functions
M = length(Functions);
Maximum = 1;

RefPoints = 100;
f1_vals = zeros(1, RefPoints);
f2_vals = zeros(1, RefPoints);

wvals = linspace(1e-5, Maximum-1e-5, RefPoints);
zvals = linspace(1e-5, Maximum-1e-5, RefPoints);

rng(128);

for i = 1:RefPoints
    z = [zvals(i) Maximum-zvals(i)];  % Initial decision vector
    w = [0.5 0.5];
    w = w / norm(w); % Normalize `w'

    assert (length(z) == M);
    assert (length(w) == M);

    % Non-Linear Conditions for ASF
    C1 = @(x) ASFCondition(x(1:N), Functions{1}, z(1), w(1));
    C2 = @(x) ASFCondition(x(1:N), Functions{2}, z(2), w(2));

    % Final Objective function
    Objective = @(x) ASF(x, Functions, M, z, w);

    % Bounds
    L = zeros(1, N+1);
    U =  ones(1, N+1) * Maximum;

    x0 = [rand(1, N) * Maximum 0.9999];

    fprintf("Iter [%2d] Solving ...\n", i);
    options = optimoptions('fmincon', 'Algorithm', 'sqp', 'TolFun', 1e-9, 'TolX',
1e-9, 'MaxFunctionEvaluations', 1e5, 'Display', 'None');
    [x, fval, exitflag, output] = fmincon(Objective, x0, [], [], [], [], L, U,
@(x)Constraint(x, C1, C2), options);
    f1_vals(i) = f1(x(1:end-1));
    f2_vals(i) = f2(x(1:end-1));
end
```

```
Iter [ 1] Solving ...
Iter [ 2] Solving ...
```

```
Iter [ 3] Solving ...
Iter [ 4] Solving ...
Iter [ 5] Solving ...
Iter [ 6] Solving ...
Iter [ 7] Solving ...
Iter [ 8] Solving ...
Iter [ 9] Solving ...
Iter [10] Solving ...
Iter [11] Solving ...
Iter [12] Solving ...
Iter [13] Solving ...
Iter [14] Solving ...
Iter [15] Solving ...
Iter [16] Solving ...
Iter [17] Solving ...
Iter [18] Solving ...
Iter [19] Solving ...
Iter [20] Solving ...
Iter [21] Solving ...
Iter [22] Solving ...
Iter [23] Solving ...
Iter [24] Solving ...
Iter [25] Solving ...
Iter [26] Solving ...
Iter [27] Solving ...
Iter [28] Solving ...
Iter [29] Solving ...
Iter [30] Solving ...
Iter [31] Solving ...
Iter [32] Solving ...
Iter [33] Solving ...
Iter [34] Solving ...
Iter [35] Solving ...
Iter [36] Solving ...
Iter [37] Solving ...
Iter [38] Solving ...
Iter [39] Solving ...
Iter [40] Solving ...
Iter [41] Solving ...
Iter [42] Solving ...
Iter [43] Solving ...
Iter [44] Solving ...
Iter [45] Solving ...
Iter [46] Solving ...
Iter [47] Solving ...
Iter [48] Solving ...
Iter [49] Solving ...
Iter [50] Solving ...
Iter [51] Solving ...
Iter [52] Solving ...
Iter [53] Solving ...
Iter [54] Solving ...
Iter [55] Solving ...
Iter [56] Solving ...
Iter [57] Solving ...
Iter [58] Solving ...
Iter [59] Solving ...
Iter [60] Solving ...
Iter [61] Solving ...
Iter [62] Solving ...
Iter [63] Solving ...
Iter [64] Solving ...
Iter [65] Solving ...
Iter [66] Solving ...
```

```
Iter [67] Solving ...
Iter [68] Solving ...
Iter [69] Solving ...
Iter [70] Solving ...
Iter [71] Solving ...
Iter [72] Solving ...
Iter [73] Solving ...
Iter [74] Solving ...
Iter [75] Solving ...
Iter [76] Solving ...
Iter [77] Solving ...
Iter [78] Solving ...
Iter [79] Solving ...
Iter [80] Solving ...
Iter [81] Solving ...
Iter [82] Solving ...
Iter [83] Solving ...
Iter [84] Solving ...
Iter [85] Solving ...
Iter [86] Solving ...
Iter [87] Solving ...
Iter [88] Solving ...
Iter [89] Solving ...
Iter [90] Solving ...
Iter [91] Solving ...
Iter [92] Solving ...
Iter [93] Solving ...
Iter [94] Solving ...
Iter [95] Solving ...
Iter [96] Solving ...
Iter [97] Solving ...
Iter [98] Solving ...
Iter [99] Solving ...
Iter [100] Solving ...
```

```
figure;
hold on;
scatter(f1_vals, f2_vals);
plot(f1_vals, f2_vals);
hold off;
xlim([0 1]);
xlabel("f_1")
ylim([0 1]);
ylabel("f_2");
grid on;
title(strcat("ZDT1 points:", num2str(RefPoints)))
legend('Evaluated Points', 'Pareto front')
```



Functions

```
function ret = f1(x)
    ret = x(1);
end

function ret = f2(x)
    G = 1 + sum(x(2:end));
    ret = G * (1 - sqrt(x(1) ./ G));
end

function ret = ASFCondition(x, Fn, z, w)
    ret = (Fn(x) - z) ./ w - x(end);
end

function [c, ceq] = Constraint(x, C1, C2)
    c = [C1(x); C2(x)];
    ceq = [];
end

% Handles alpha
function ret = ASF(x, Functions, M, z, w)
    C = zeros([1 M]);

    x = x(1:end-1);  % alpha is not required
```

```
    for i = 1:M
        C(i) = (Functions{i}(x) - z(i)) ./ w(i);
    end
    ret = max(C);
end
```