# Techniques in Operations Research

# Assignment 1 – 2018

Christian Drozdowicz, 832391

March 21, 2018

## Question 1: Let $f(x) = 8e^{1-x} + 7 \times \log(x)$

## Part A)

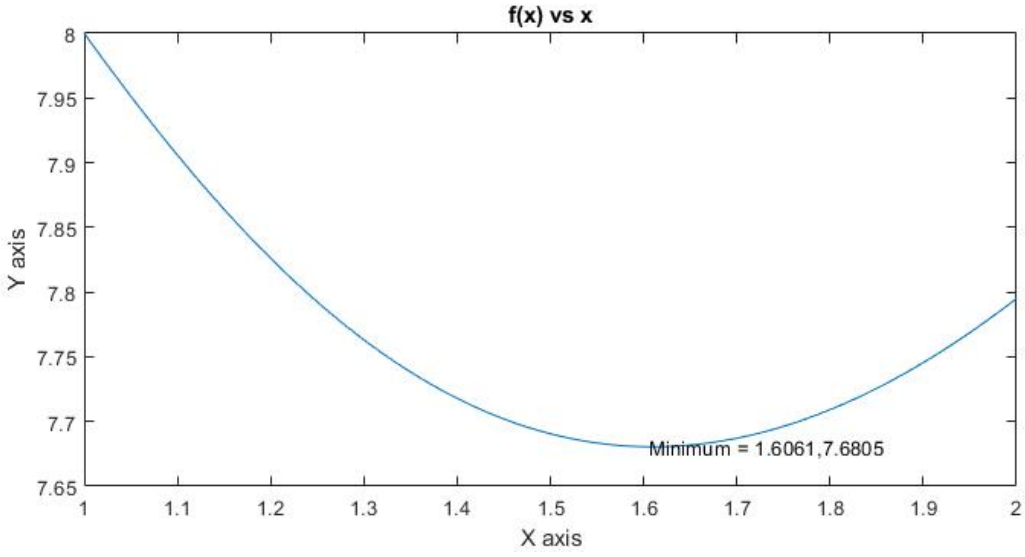
Unimodal, by definition, is a function *f* that is continuous on its domain [a,b] that only has one local minimum.

To judge whether our $f(x)$ is unimodal on the interval [a,b]=[1,2], we shall plot $f(x)$ versus $x$ over this given interval.

MATLAB CODE:

```
x = linspace(1,2);
y = (8*exp(1-x)+7*log(x));
    indexmin = find(min(y) == y);
    xmin = x(indexmin);
    ymin = y(indexmin);
plot(x,y)
strmin = ['Minimum = ',num2str(xmin) ',' num2str(ymin)];
text(xmin,ymin,strmin,'HorizontalAlignment','left');
```

PLOT



As we can see by inspection, there only occurs one local minimum on the interval [1,2] and the function is continuous on this domain, this is further confirmed by the Matlab program to find the local minimum of this function and plot the function. Thus we can conclude that $f(x)$ is unimodal on the interval [1,2].

## Part B)

Now we shall go through the Fibonacci Search Method to locate the minimizer $x^*$ of *f* over [1,2] to within an uncertainty of 0.08:

First we note that $f'(x) = \frac{7}{x} - 8(e^{1-x}) \Rightarrow f'(x) = 0 \Rightarrow x_{min} = 1.61$

**Step 1**: solve for *n*, $\frac{2-1}{\mathrm{F}_n} < 0.16$ That is, $F_n > \frac{1}{0.16} = 6.25 \Rightarrow F_n = 8 \Rightarrow n = 5$ .

Therefore, we require 5 *f*-calculations.

**Step 2**: *k*=5

$$\mathrm{p}= 2 - \frac{5}{8}(2-1) = 1.375$$

$$\mathrm{q}= 1 + \frac{5}{8}(2-1) = 1.625$$

$$f(1.375) = 7.72749$$

$$f(1.625) = 7.68065$$

**Step 3**: *k*=4, $f(1.375) > f(1.625)$

$$a = 1.375$$

$$p = 1.625$$

$$\mathrm{q}= 1.375 + \frac{3}{5}(2-1.375) = 1.75$$

$$b = 2$$

$$f(1.625) = 7.68065$$

$$f(1.75) = 7.69624$$

**Step 4**: *k*=3, $f(1.625) < f(1.75)$

$$a = 1.375$$

$$\mathrm{p}= 1.75 - \frac{2}{3}(1.75 - 1.375) = 1.5$$

$$q = 1.625$$

$$b = 1.75$$

$$f(1.5) = 7.6905$$

$$f(1.625) = 7.68065$$

**Step 5**: *k*=2, $f(1.5) > f(1.625)$

$$a = 1.5$$

$$p = 1.625$$

$$\text{q} = 1.5 + 2(0.08) = 1.66$$

$$b = 1.75$$

$$f(1.625) = 7.68065$$

$$f(1.66) = 7.68253$$

**Step 6**: *k*=1, $f(1.625) < f(1.66)$

$$a = 1.5$$

$$b = 1.66$$

Thus, $x_{min} \in [1.5, 1.66]$, that is $x_{min} = 1.58 \pm 0.08$

Which follows the output for this function on the interval [1,2] with tolerance 0.08 on the provided Matlab Fibonacci Search program.

fibonacciSearch(@(x) 8*exp(1-x)+7*log(x),1,2,0.08)

ans = 'using the first 5 numbers of the fibonacci sequence'

Iter =

| | | | | |
|---|---|---|---|---|
| 1.0000 | 1.0000 | 2.0000 | 8.0000 | 7.7951 |
| 3.0000 | 1.3750 | 2.0000 | 7.7275 | 7.7951 |
| 2.0000 | 1.3750 | 1.7500 | 7.7275 | 7.6962 |
| 4.0000 | 1.5000 | 1.7500 | 7.6905 | 7.6962 |
| 5.0000 | 1.5000 | 1.6600 | 7.6905 | 7.6825 |

ans = 1.5800

Which is the same result.

**Qu : Let $f(x) = (40$ $\times \log(40x + 1) - 200x$**

We mine the minimizer $x$ ng Golden Section Search through the inte l $[0, \infty$ olerance of 0.3.

First hat $f'(x) = 40\log(4$ $- 160 \Rightarrow f'(x) = 0 \Rightarrow x_{min} = 1.33995$

To fir is algorithm, we mus ate an upper bound for the location of the minir lo this for our functio is a continuous, unimodal function on $[0, \infty)$, we ne d a point *b* such that $x_{min} < b$.

**Step** small initial increment value *T*, thus let *T*=1

$$k = 1$$

$$p = 0$$

$$q = T = 1$$

$$f(p) = 0$$

$$f(q) = -47.7435$$

**Step** $f(q)$

$$k = 2$$

$$p = 1$$

$$q = p + 2^{k-1}T = 1 + 2(1) = 3$$

$$f(p) = -47.7435$$

$$f(q) = -19.7093$$

Now $(q)$, so we stop.

Thus $x_{min} < 3$ with *a*=0 and *b*=3.

## Golden Section Search:

Now we shall go through the Golden Section Search Method to locate the minimizer $x^*$ of $f$ over [0,3] to within an uncertainty of 0.3:

We notice that the golden ratio is given by $\gamma = \frac{\sqrt{5}-1}{2} \approx 0.618$

**Step 1**: *k*=1

$$a = 0$$

$$\text{p}= b - \gamma(b-a) = 3 - \gamma(3-0) = 1.1459$$

$$\text{q}= a + \gamma(b-a) = 0 + \gamma(3-0) = 1.8541$$

$$f(1.1459) = -49.0182$$

$$f(1.8541) = -46.1361$$

**Step 2**: *k*=2, $f(1.1459) < f(1.8541)$

$$a = 0$$

$$p = b - \gamma(b-a) = 1.18541 - \gamma(1.8541-0) = 0.7082$$

$$\text{q}= p = 1.1459$$

$$b = q = 1.8541$$

$$f(0.7082) = -42.5542$$

$$f(1.1459) = -49.0182$$

**Step 3**: *k*=3, $f(0.7082) > f(1.1459)$

$$a = p = 0.7082$$

$$p = q = 1.1459$$

$$\text{q}= a + \gamma(b-a) = 0.7082 + \gamma(1.8541-0.7082) = 1.4164$$

$$b = 1.8541$$

$$f(1.1459) = -49.0182$$

$$f(1.4164) = -49.5141$$

**Step 4**: *k*=4, $f(1.1459) > f(1.4164)$

$$a = 1.1459$$

$$p = 1.4164$$

$$\text{q} = 1.1459 + \gamma(1.8541 - 1.1459) = 1.5836$$

$$b = 1.8541$$

$$f(1.4164) = -49.5141$$

$$f(1.5836) = -48.776$$

**Step 5**: *k*=5, $f(1.4164) < f(1.5836)$

$$a = 1.1459$$

$$p = 1.5836 - \gamma(1.5836 - 1.1459) = 1.3131$$

$$\text{q} = 1.4164$$

$$b = 1.5836$$

$$f(1.3131) = -49.5875$$

$$f(1.4164) = -49.5141$$

However, we now notice that $(b - a) = (1.5836 - 1.1459) = 0.4377 < 2 \times 0.3 = 2\varepsilon$

So, we STOP.

Therefore, the final interval is $[1.1459, 1.5836]$ and the final estimate is $1.3648 \pm 0.3$

Which is the same result as the provided Golden Section Search Matlab program:

```
>> goldenSectionSearch(@(x) (40*x+1)*log(40*x+1)-200*x,0,3,0.3)
Iter =

   1.0000         0    3.0000      -Inf    8.7730
   2.0000         0    1.8541      -Inf    7.7271
   3.0000    0.7082    1.8541    8.2955    7.7271
   4.0000    1.1459    1.8541    7.8673    7.7271
   5.0000    1.1459    1.5836    7.8673    7.6810
```

## Question 3: Let $f(x) = (2x-1)^3 \quad (4-10x)^4$

We first need to find an interval that contains nimizer of this function. To start we need to obtain a suitable step size to begin to r this interval. To do this we shall use the Armijo-Goldstein and Wolff conditions.

For a unimodal, continuous, and differentiable on *f(x)*

The input for the initial variables are:

*T*=0.06, $\sigma$=0.3, $\mu$=0.5

**Step 1**:

$t_{lo}$

$t_{hi}$ =

t= $T$ =

$f(t) = f(0.0$ 33.853

$f(0) + t\sigma f'(0) = 1023 + (0.0$ $\times (-10234)) = 838.788$

Since $f(t) < f(0) + t\sigma f'(0)$, the -Goldstein condition is met.

Now:

$f'(t) = f'(0.0$ ·6283.99

$\mu f'(0) = 0.5 \times (-$ $4) = -5117$

Since $f'(t) < \mu f'(0)$ Wolff conditio t met so we proceed to step 2.

**Step 2:** $f'(t) < \mu f'(0)$

$t_{lo} = t$ 6

Since $t_{hi} = \infty \Rightarrow$ t $= 2t$

$$f(t) = f(0.12) = 245.423$$

$$f(0) + t\sigma f'(0) = 1023 + (0.12 \times 0.3 \times (-10234)) = 654.576$$

Since $f(t) < f(0) + t\sigma f'(0)$, the Armijo-Goldstein condition is met.

Now:

$$f'(t) = f'(0.12) = -3508.85$$

$$\mu f'(0) = 0.5 \times (-10234) = -5117$$

Since $f'(t) > \mu f'(0)$ Wolff condition is met so we stop.

Since both Armijo-Goldstein and Wolff conditions are met, we stop the algorithm and take the step size: *t*=0.12

**Now we will use the step size previously found to bind the function on the Real Number line $\mathbb{R}$**

**Step 1: Choose increment T=0.12 from previous algorithm**

$$k = 1$$

$$p = 0$$

$$q = 0.12$$

$$f(p) = f(0) = 1023$$

$$f(q) = f(0.12) = 245.423$$

**Step 2:** $f(p) > f(q)$

$$k = 2$$

$$p = 0.12$$

$$q = kT = 2 * 0.12 = 0.24$$

$$f(p) = f(0.12) = 245.423$$

$$f(q) = f(0.24) = 26.074$$

**Step 2:** $f(p) > f(q)$

$$k = 3$$

$$p = 0.24$$

$$q = kT = 3 * 0.12 = 0.36$$

$$f(p) = f(0.24) = 26.074$$

$$f(q) = f(0.36) = 0.080$$

**Step 2:** $f(p) > f(q)$

$$k = 4$$

$$p = 0.36$$

$$q = kT = 4 * 0.12 = 0.48$$

$$f(p) = f(0.36) = 0.080$$

$$f(q) = f(0.48) = 1.638$$

Since $f(p) < f(q)$, we stop.

The final interval to ensure we include the minimiser is $a = 0.24, b = 0.48$

Now to see the program:

```
%This function runs the method of false position to find the final estimate
%of the minimiser of the provided function within the tolerance given

%It assumes the the func is unimodal and continuous on the interval [a,b]


function false_position(functiontominimise,a,b,tolerance)

if (nargin ~= 4)     % in Matlab the operator ~= means "not equal to"
    error('four input arguments are required')
end

% Check that input parameters have appropriate values

    if(b <= a)
        error('b must be strictly greater than a')
    end

    if(tolerance <= 0)
        error('tolerance must be strictly positive')
    end

% begin the Method of False Position

% Set all initial parameters and variables with correct equations
% and conditions

k = 1;

ga = feval(functiontominimise, a);
gb = feval(functiontominimise, b);

p = a + ((ga * (b-a))/(ga-gb));
gp = feval(functiontominimise, p);
c = gp;
absolute_c = abs(c);

% Set iteration
Iter = [k, a,b,p,gp];

% Set while loop to stop when final condition is met
while (absolute_c > tolerance)
    k=k+1;


    if gp < 0 %first condition
        a = p;
        ga = feval(functiontominimise, a);
        p = a + ((ga * (b-a))/(ga-gb));
        %calculate gp
        gp = feval(functiontominimise, p);
        c = gp;
        absolute c = abs(c); %if absolute c > tolerance continue
```

```
    else %everything else that is not gp<0
        b = p;
        gb = feval(functiontominimise, b);
        p = a + ((ga * (b-a))/(ga-gb));
        %calculate gp
        gp = feval(functiontominimise, p);
        c = gp;
        absolute_c = abs(c);%if absolute_c > tolerance continues

    end


%once condition is met, end the program
end

%print out number of iterations, the final estimate, and the value of g

fprintf('It took %.0f iterations, the finals estimate is %.5f, and the value
of g is %.3e\n',k,p,gp)
```

Running this program with the conditions we have previously found, then we run this function:

>> false_position(@(x) 160000*x^3-191976*x^2+76776*x-10234,0.24,0.48,0.00001)

Which returns the following output:

It took 1068 iterations, the finals estimate is 0.38763, and the value of g is 9.850e-06

## Question 3 Discussion

If we notice that $f'(x) = 160000x^3 - 191976x^2 + 76776x - 10234 \Rightarrow f'(x) = 0 \Rightarrow x_{min} = 0.387627$

Therefore, the program is accurate to at least 5 decimal places to find the minimum of the original function.

However, if we were to know where the minimiser was supposed to lie with more initial accuracy, instead our interval could have been from [0.36,0.48] from the Armijo-Goldstein and Wolff conditions. To which if we run the program now changing the input *a* for the lower bound:

>> false_position(@(x) 160000*x^3-191976*x^2+76776*x-10234,0.36,0.48,0.00001)

It took 132 iterations, the finals estimate is 0.38763, and the value of g is -9.840e-06

This took only 132 iterations as compared to 1068 of the original, vastly speeding up the process.