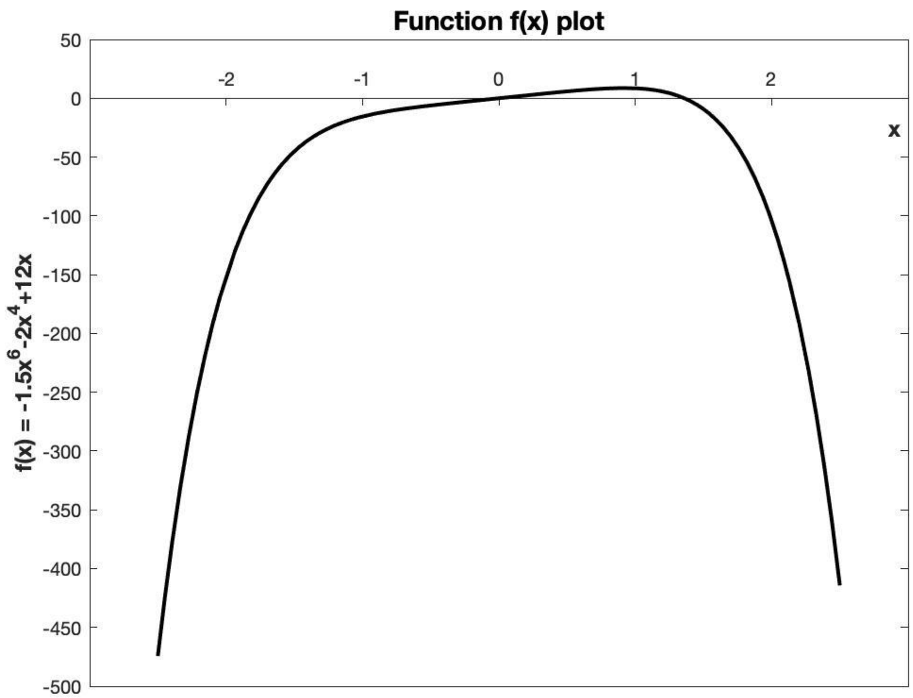
Robert Heeter
BIOE 391 Numerical Methods – Due 13 February 2022

Problem Set 4 Solutions

1.

a. Figure



b. See attached handwritten work at end of document. $f''(x) \leq 0$ for all real values of x, making $f(x)$ concave (concave down).

c. Output
Bisection method output:
maximum (x_max, f(x_max)) = (0.916912, 8.697930)
approx. relative error = 0.000416
iterations = 20

2. Output
Golden-section method output:
time at maximum (s) = 3.831523
maximum altitude (m) = 192.860863
approx. relative error = 0.021326
iterations = 20

3. Output
Parabolic interpolation output:
minimum (x_min, f(x_min)) = (1.427552, -1.775726)
approx. relative error = 0.000026
iterations = 7

Robert Heeter
BIOE 391 Numerical Methods – Due 13 February 2022

4. **Output**
   Fminbnd output:
   minimum of cost function (x_A, cost) = (0.587699, 11.149510)

5.
   a. **Output**
      Fminbnd output:
      minimum of drag function (V_min, D_min) = (509.818120, 3118.974190)

   b. **Output**
      Fminbnd output:

W:	V_min:	D_min:
12000	441.5154	2339.2306
13000	459.5438	2534.1665
14000	476.8912	2729.1024
15000	493.6293	2924.0383
16000	509.8181	3118.9742
17000	525.5085	3313.9101
18000	540.7438	3508.8460
19000	555.5614	3703.7819
20000	569.9940	3898.7177

6. **Output**
   Fminbnd output:
   optimum velocity (m/s) = 248.586834
   drag force (N) = 45938.710896
   ratio drag force to velocity = 184.799453

Robert Heeter
BIOE 391 Numerical Methods – Due 13 February 2022

Complete MATLAB Code

```
% Robert Heeter
% BIOE 391 Numerical Methods
% HOMEWORK 4 MATLAB SCRIPT

clc, clf, clear, close all

%% P1. PROBLEM 7.4
disp('P1. PROBLEM 7.4');

% Plot function (PART A)
fx = @(x) (-1.5.*(x.^6))-(2.*(x.^4))+(12.*x); % function f(x)
figure
fplot(fx,[-2.5, 2.5],'-k','LineWidth',2);
xlabel('x','FontSize',12,'FontWeight','bold');
ylabel('f(x) = -1.5x^6-2x^4+12x','FontSize',12,'FontWeight','bold');
title('Function f(x) plot','FontSize',14,'FontWeight','bold');
ax = gca;
ax.XAxisLocation = 'origin';
axis([-3 3 -500 50]);

% Maximum value (PART C)
dfdx = @(x) (-9.*(x.^5))-(8.*(x.^3))+(12); % derivative of f(x)
[x_max,~,ea,iter] = bisection(dfdx,-2,2,0,20); % use bisection function (below)
fx_max = fx(x_max); % evaluate f(x) at x_max
disp('Bisection method output:') % display results
fprintf('maximum (x_max, f(x_max)) = (%f, %f)\napprox. relative error = %f\niterations =
%d\n\n',x_max,fx_max,ea,iter);


%% P2. PROBLEM 7.17
disp('P2. PROBLEM 7.17');

g = 9.81; % gravitational acceleration (m/s^2)
z_0 = 100; % initial altitude (m)
v_0 = 55; % initial velocity (m/s)
m = 80; % mass (kg)
c = 15; % linear drag coefficient (kg/s)

z = @(t) z_0 + ((m/c)*(v_0+(m*g/c))*(1-exp(-1*(c/m)*t)))-(m*g*t/c); % motion of bungee jumper function
[t,z_max,ea,iter] = goldensectionmax(z,0,20,0,20); % use golden-section function (below)

% Display results
disp('Golden-section method output:')
fprintf('time at maximum (s) = %f\nmaximum altitude (m) = %f\napprox. relative error = %f\niterations =
%d\n\n',t,z_max,ea,iter);


%% P3. PROBLEM 7.19
disp('P3. PROBLEM 7.19');

f = @(x) ((x^2)/10)-(2*sin(x)); % function f(x)
xl = 0; % initial guesses
xu = 4;

[x_min,fx_min,ea,iter] = parabolicinterpolationmin(f,xl,xu,0.001,10); % use parabolic interpolation
function (below)

% Display results
disp('Parabolic interpolation output:')
fprintf('minimum (x_min, f(x_min)) = (%f, %f)\napprox. relative error = %f\niterations =
%d\n\n',x_min,fx_min,ea,iter);
```

Robert Heeter
BIOE 391 Numerical Methods – Due 13 February 2022

```
%% P4. PROBLEM 7.29
disp('P4. PROBLEM 7.29');

C = 1; % proportionality constant (assume basis of C = 1)
cost = @(x) C*((1./((1-x).^2)).^0.6)+(6*((1./x).^0.6)); % cost function
[x,fx] = fminbnd(cost,0,1); % use fminbnd function

% Display results
disp('Fminbnd output:')
fprintf('minimum of cost function (x_A, cost) = (%f, %f)\n\n',x,fx);


%% P5. PROBLEM 7.35
disp('P5. PROBLEM 7.35');

% PART A
sigma = 0.6; % ratio of air density between flight altitude and sea level
W = 16000; % weight
D = @(V) (0.01*sigma*(V^2)) + ((0.95/sigma)*((W/V)^2)); % drag function
[V_mina,D_mina] = fminbnd(D,0,5000); % use fminbnd function
disp('Fminbnd output:') % display results
fprintf('minimum of drag function (V_min, D_min) = (%f, %f)\n\n',V_mina,D_mina);

% Sensitivity analysis (PART B)
sigma = 0.6; % ratio of air density between flight altitude and sea level
W_range = (12000:1000:20000)'; % weight range
V_minb = zeros(size(W_range)); % preallocate result vectors
D_minb = zeros(size(W_range));

for i = 1:length(W_range)
    W = W_range(i);
    D = @(V) (0.01*sigma*(V^2)) + ((0.95/sigma)*((W/V)^2)); % drag function
    [V_minb(i),D_minb(i)] = fminbnd(D,0,5000); % use fminbnd function
end

disp('Fminbnd output:') % display results
fprintf(' W:      V_min:       D_min: \n');
fprintf(' %5.0f    %7.4f      %8.4f\n',[W_range,V_minb,D_minb]');
disp(' ');


%% P6. PROBLEM 7.42
disp('P6. PROBLEM 7.42');

W = 670000; % weight (N)
A = 150; % wing platform area (m^2)
AR = 6.5; % aspect ratio
C_D0 = 0.018; % drag coefficent at zero lift
rho = 0.413; % air desnity (kg/m^3)

C_L = @(v) (2*W)/(rho*(v^2)*A); % lift equation
C_D = @(v) C_D0 + (((C_L(v))^2)/(pi*AR)); % drag equation
F_D = @(v) W*(C_D(v)/C_L(v)); % drag force equation

[v,ratio] = fminbnd(@(v) F_D(v)/v,0,500); % use fminbnd function to find minimum ratio of drag force to
velocity
disp('Fminbnd output:') % display results
fprintf('optimum velocity (m/s) = %f\ndrag force (N) = %f\nratio drag force to velocity =
%f\n\n',v,F_D(v),ratio);


%% Additional Functions

function [root,fx,ea,iter] = bisection(func,xl,xu,es,maxit,varargin)
% ABOUT: Bisection method for finding roots, adapted from textbook .m file.
% INPUTS: func = function; xl, xu = lower and upper bounds; es = desired
```

Robert Heeter
BIOE 391 Numerical Methods – Due 13 February 2022

```
% relative error (as percent); maxit = maximum iterations
% OUTPUTS: root = real root; fx = function value at root; ea = approximate
% relative error (as percent); iter = number of iterations

if nargin < 3
    error('At least 3 input arguments required.')
end
test = func(xl,varargin{:})*func(xu,varargin{:});
if test > 0
    error('No sign change.')
end
if nargin < 4 || isempty(es)
    es = 0.0001;
end
if nargin < 5 || isempty(maxit)
    maxit = 50;
end

iter = 0;
xr = xl;
ea = 100;
while (1)
    xrold = xr;
    xr = (xl + xu)/2;
    iter = iter + 1;
    if xr ~= 0
        ea = abs((xr - xrold)/xr) * 100;
    end
    test = func(xl,varargin{:})*func(xr,varargin{:});
    if test < 0
        xu = xr;
    elseif test > 0
        xl = xr;
    else
        ea = 0;
    end
    if ea <= es || iter >= maxit
        break
    end
end

root = xr;
fx = func(xr, varargin{:});

end


function [x,fx,ea,iter] = goldensectionmax(f,xl,xu,es,maxit,varargin)
% ABOUT: Golden-section method for finding maximums, adapted from textbook
% .m file.
% INPUTS: f = function; xl, xu = lower and upper bounds; es = desired
% relative error (as percent); maxit = maximum iterations
% OUTPUTS: x = location of maximum; fx = function value at maximum; ea =
% approximate relative error (as percent); iter = number of iterations

if nargin < 3
    error('At least 3 input arguments required.')
end
if nargin < 4 || isempty(es)
    es = 0.0001;
end
if nargin < 5 || isempty(maxit)
    maxit = 50;
end

phi=(1+sqrt(5))/2;
```

```
iter = 0;
d = (phi-1) * (xu-xl);
x1 = xl + d;
x2 = xu - d;
f1 = f(x1,varargin{:});
f2 = f(x2,varargin{:});
while(1)
    xint = xu-xl;
    if f1 > f2 % changed condition to f1 > f2 to identify a maximum instead of minimum
        xopt = x1;
        xl = x2;
        x2 = x1;
        f2 = f1;
        x1 = xl + (phi-1) * (xu-xl);
        f1 = f(x1,varargin{:});
    else
        xopt = x2;
        xu = x1;
        x1 = x2;
        f1 = f2;
        x2 = xu - (phi-1) * (xu-xl);
        f2 = f(x2,varargin{:});
    end
    iter = iter+1;
    if xopt ~= 0
        ea = (2-phi) * abs(xint/xopt)*100;
    end
    if ea <= es || iter >= maxit
        break
    end
end

x = xopt;
fx = f(xopt,varargin{:});

end


function [x,fx,ea,iter] = parabolicinterpolationmin(f,xl,xu,es,maxit,varargin)
% ABOUT: Parabolic interpolation method for finding minimums.
% INPUTS: f = function; xl, xu = lower and upper guesses; es = desired
% relative error (as percent); maxit = maximum iterations
% OUTPUTS: x = location of minimum; fx = function value at minimum; ea =
% approximate relative error (as percent); iter = number of iterations

if nargin < 3
    error('At least 3 input arguments required.')
end
if xl >= xu
    error('Check lower and upper guesses.')
end

xm = xl + (xu-xl)/2; % midpoint of initial interval
x1 = xl;
x2 = xm;
x3 = xu;
if f(x2,varargin{:}) >= f(x1,varargin{:}) || f(x2,varargin{:}) >= f(x3,varargin{:}) % check whether
guesses bracket minimum
    error('Guesses do not bracket minimum.')
end

if nargin < 4 || isempty(es)
    es = 0.0001;
end
if nargin < 5 || isempty(maxit)
    maxit = 50;
```

```
end

iter = 0;
while(1)
    % use formula for parabolic fit
    xn_numerator = (((x2-x1)^2)*(f(x2,varargin{:})-f(x3,varargin{:}))) -
(((x2-x3)^2)*(f(x2,varargin{:})-f(x1,varargin{:})));
    xn_denominator = ((x2-x1)*(f(x2,varargin{:})-f(x3,varargin{:}))) -
((x2-x3)*(f(x2,varargin{:})-f(x1,varargin{:})));
    xn = x2 - (0.5)*(xn_numerator/xn_denominator);
    iter = iter+1;
    if xn ~= 0
        ea = abs((xn - x2)/xn) * 100;
    end
    if ea <= es || iter >= maxit
        break
    end
    % adjust points for parabolic fit and iterate
    if xn < x2
        x1 = x2;
        x2 = xn;
    else
        x3 = x2;
        x2 = xn;
    end
end

x = xn;
fx = f(xn,varargin{:});

end
```

Robert Heeter
BIOE 391
2/13/22

Problem Set # 4

1\. b. $f(x) = -1.5x^6 - 2x^4 + 12x$

$f'(x) = -9x^5 - 8x^3 + 12$ ← first derivative

$f''(x) = -45x^4 - 24x^2$ ← second derivative

Because $f''(x) \leq 0$ for all real values of x, $f(x)$ is concave. x^4 and x^2 are always positive for any real number x, so $-45x^4 - 24x^2$ is always negative (or zero).