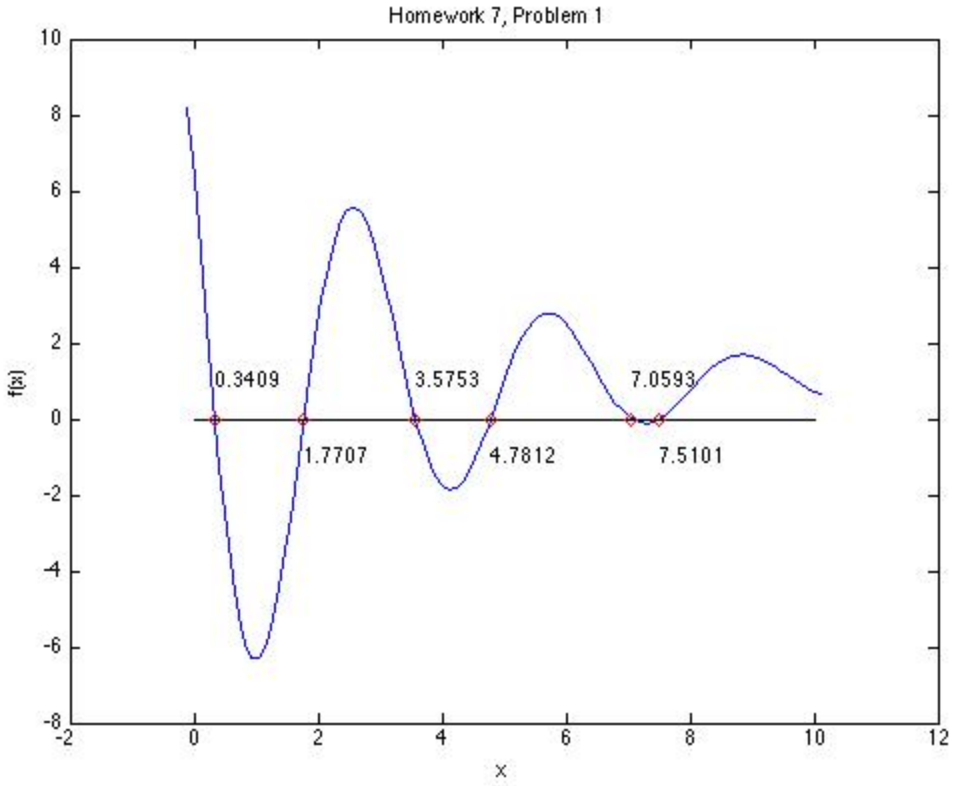
Homework 07 - Steve Mazza

Table of Contents

Problem 1

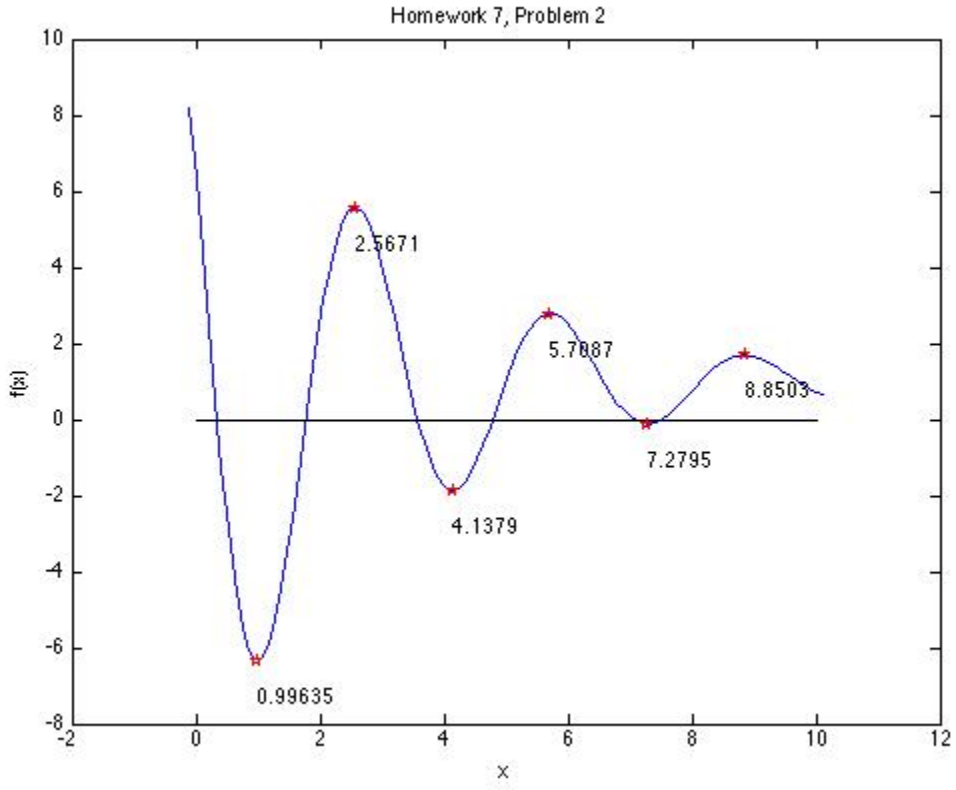
```
clear all; clc; close all;

xmin = 0;
xmax = 10;

% Create function handle.
myFun = @(x) 10*exp(-0.3*x).*cos(2*x+1)+1;

% Bracket zeros within the interval 0..10 with lroot.m
Br=lroot(myFun,xmin,xmax,20,0);

% Pre-allocae array.
f_roots = zeros(1, numel(Br(:,1)));

% Parse Br and use fzero() to find each root.
for i = 1:numel(Br(:,1))
    f_roots(i) = fzero(myFun,[Br(i,1) Br(i,2)]);
end

% Plot function, roots, values, and markers.
xp=linspace(xmin-.1,xmax+.1);
fu1=fcnchk(myFun);
fp=feval(fu1,xp);
plot(xp,fp);
hold on;
title('Homework 7, Problem 1');
xlabel('x');
ylabel('f(x)');
line([xmin xmax],[0 0],'Color',[0 0 0]);
for i = 1:numel(f_roots)
    % Add the root to the plot.
    plot(f_roots(i),0,'rd');
    % Add numerical value.
    text(f_roots(i),-(-1)^i,num2str(f_roots(i)));
end
```



Problem 2

```
clear all; clc; close all;

syms x;
xmin = 0;
xmax = 10;

% Create function and handle.
fun = 10*exp(-0.3*x).*cos(2*x+1)+1;
fun_h = @(x) 10*exp(-0.3*x).*cos(2*x+1)+1;

% Compute first order derivative and convert to inline function.
d_fun = inline(diff(fun));

% User lroot() to find zeros.
Br=lroot(d_fun,xmin,xmax,20,0);

% Pre-allocae array.
f_m = zeros(1, numel(Br(:,1)));

% Parse Br and use fzero() to find each root.
for i = 1:numel(Br(:,1))
    f_roots(i) = fzero(d_fun,[Br(i,1) Br(i,2)]);
end
```

```
% Plot original function, show minimums, maximums.
xp=linspace(xmin-.1,xmax+.1);
fp=feval(inline(fun),xp);
plot(xp,fp);
hold on;
title('Homework 7, Problem 2');
xlabel('x');
ylabel('f(x)');
line([xmin xmax],[0 0],'Color',[0 0 0]);
for i = 1:numel(f_roots)
    % Add the min or max to the plot.
    plot(f_roots(i),fun_h(f_roots(i)),'rp');
    % Add numerical value.
    text(f_roots(i),fun_h(f_roots(i))-1,num2str(f_roots(i)));
end
```

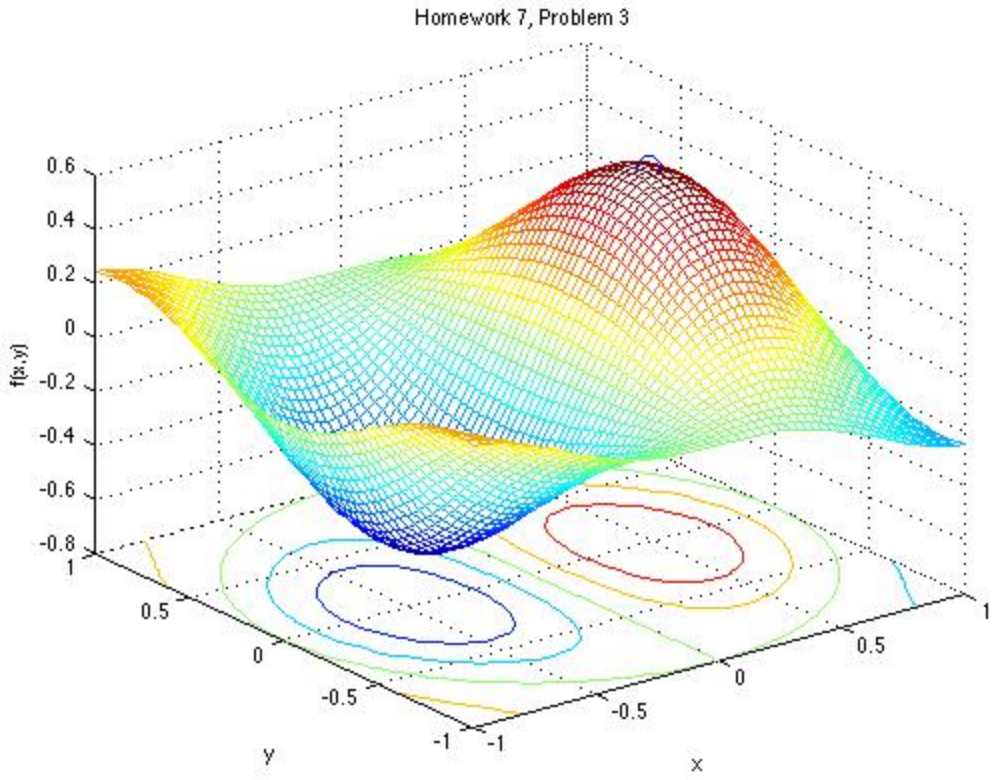


Problem 3

```
clear all; clc; close all;
warning off;

xmin = -1;
xmax = 1;

fun_h = @(x) (-x(1)/(sqrt(x(1)^2+x(2)^2)))* ...
```

```
    besselj(1,3.8316*sqrt(x(1)^2+x(2)^2));

fun_xy = @(x,y) (x/(sqrt(x^2+y^2)))* ...
    besselj(1,3.8316*sqrt(x^2+y^2));

% The following two calls fail!!!
[x0,fval0,exitflag0,output0] = fminsearch(fun_h,[0.5; -0.5]);
[x1,fval1,exitflag1,output1] = fminunc(fun_h,[0.5; -0.5]);

fprintf('\n\nfminsearch() took %d iterations and fminumc() took %d.\n' ...
    ,output0.iterations,output1.iterations);

% Plot and add a maximum.
ezmeshc(fun_xy,[xmin,xmax]);
hold on;
title('Homework 7, Problem 3');
xlabel('x');ylabel('y');zlabel('f(x,y)');
plot3(x1(1),x1(2),-1*fval1,'ob','MarkerSize',12);
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the default value of the function tolerance.



fminsearch() took 35 iterations and fminumc() took 7.
```

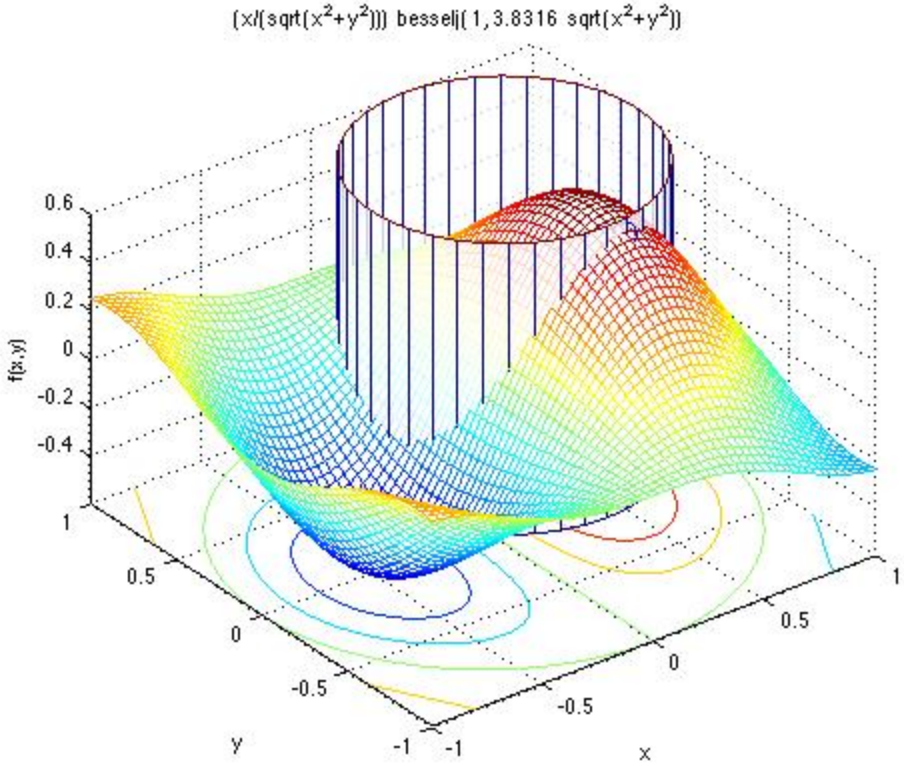


Problem 4

```
clear all; clc; close all;
warning off;

% Define the function.
fun_h = @(x) (-x(1)/(sqrt(x(1)^2+x(2)^2)))* ...
    besselj(1,3.8316*sqrt(x(1)^2+x(2)^2));

fun_xy = @(x,y) (x/(sqrt(x^2+y^2)))* ...
    besselj(1,3.8316*sqrt(x^2+y^2));

% Define the non-linear constraints.
c = @(x) 0.6^2 - (x(1) - 0.4)^2 - (x(2) - 0.4)^2;
ceq = @(x) [];
nonlinfcn = @(x)deal(c(x),ceq(x));

% Define the other arguments to fmincon().
fun = fun_h;        % function to evaluate.
x0 = [0.5;0];       % starting point for x.
A = [];             % inequality matrix.
b = [];             % inequality vector.
Aeq = [];           % equality matrix.
beq = [];           % eqyality vector.
lb = [-1;-1];       % lower bound for x.
```

```
ub = [1;1];         % upper bound for x.

% Run the solution.
[x_,fval,exitflag,output] = fmincon(fun_h,x0,A,b,Aeq,beq,lb,ub,nonlinfcn);

% Plot the work.
[x,y,z] = cylinder(0.6,40);
mesh(x+0.4,y+0.4,1.2*z-0.6,'FaceAlpha',0.8);
axis equal;
hold on;
title('Homework 7, Problem 4');
xlabel('x');ylabel('y');zlabel('f(x,y)');
ezmeshc(fun_xy,[-1,1]);
plot3(x_(1),x_(2),-1*fval,'ob','MarkerSize',12);
```

```
        Local minimum possible. Constraints satisfied.

        fmincon stopped because the size of the current search direction is less t
        twice the default value of the step size tolerance and constraints are
        satisfied to within the default value of the constraint tolerance.



        Active inequalities (to within options.TolCon = 1e-06):
          lower      upper     ineqlin   ineqnonlin
                                                1
```



Published with MATLAB® R2013a