Kathleen Dao
Daok
AMATH 301 section A
Written Homework 5

Problem 1

a) We cannot use Newton's method in this problem because it is a derivative based method. The equation that we are trying to find a minimum for is the max(abs(eig(M))) and looking at the functions in our code, we cannot take the derivative of this with respect to w.

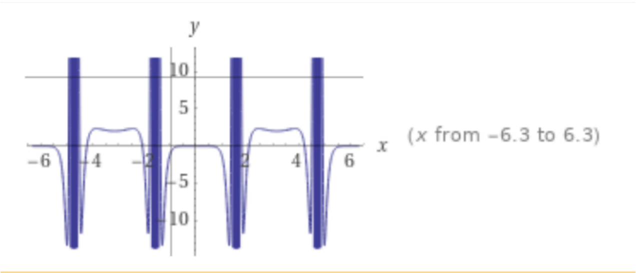
The section search function for A_114 with a=1, b=2, c=0.5001 took about 0.6 seconds.
The golden section search function for A_114 with a=1 and b=2 took about 0.4 seconds.
The section search function for A_1000 took about 81 seconds.
The golden section search function for A_1000 took about 62 seconds.

I think that the golden search method provided a substantial improvement with it taking about 75% of the time that the section search method took, which makes sense because in the golden section search, our loop calls the function f once every time around the loop and twice before the loop, whereas the section search method calls the function f twice per step through the loop. Calling a function as we know is time consuming and the difference that the golden section search method has in timing from the sections search method has been made more evident when the function f contains larger matrices.

```
        clear all; close all; clc
%Problem 1 part A

%Section Search and golden search for A_114 with a=1, b=2, c=0.5001
v1=zeros(1,114)
v1(1,:)=2;
A=diag(v1);
v2=zeros(1,113);
v2(1,:)=-1;
B=diag(v2,1);
C=diag(v2,-1);
A_114=A+B+C;

max_abs_eig_A114=@(w)(f(w, diag(diag(A_114)), tril(A_114,-1), triu(A_114,+1)))
tic
w = section_search(max_abs_eig_A114, 1, 2, 0.5001)
toc

tic
w2 = golden_section_search(max_abs_eig_A114, 1, 2)
toc


%Section search and golden search methods of A_1000
v12=zeros(1,1000)
v12(1,:)=2;
A2=diag(v12);
v22=zeros(1,999);
v22(1,:)=-1;
B2=diag(v22,1);
C2=diag(v22,-1);
A_1000=A2+B2+C2;
```

```
max_abs_eig_A1000 = @(w)(f(w, diag(diag(A_1000)), tril(A_1000, -1), triu(A_1000, +1)));
tic
w_1000 = section_search(max_abs_eig_A1000, 1, 2, 0.5001)
toc

tic
w2_1000 = golden_section_search(max_abs_eig_A1000, 1, 2)
toc
function y=f(w,D,L,U)
    P=1/w*D+L;
    T=(w-1)/w*D+U;
    M=-P\T;
    y=max(abs(eig(M)));
end
function x = section_search(f, a, b, c)
    tolerance = 1e-8;
    for k=1:1000
        x = c*a + (1-c)*b;
        y = (1-c)*a + c*b;
        if f(x) < f(y)
            b = y;
        else
            a = x;
        end
        if (b-a) < tolerance
            break;
        end
    end
end
function x = golden_section_search(f,a,b)
    c = (-1 + sqrt(5)) / 2;
    tolerance = 1e-8;
    x = c*a + (1-c)*b;
    fx = f(x);
    y = (1-c)*a + c*b;
    fy = f(y);
    for k = 1:1000
        if fx < fy
            b = y;
            y = x;
            fy = fx;
            x = c*a + (1-c)*b;
            fx = f(x);
        else
            a = x;
            x = y;
            fx = fy;
            y = (1-c)*a +c*b;
            fy = f(y);
        end
        if (b-a) < tolerance
            break;
        end
    end
end
```

b) With Newton’s method, any initial guess that was not close enough to the value of the answer I obtained from the golden search method did not result in a minimum value of the function. This is because Newton’s method finds a point where f’(x)=0. Looking at the graph of f(x) shown below, there are many points at which f’(x)=0, but are not minimum points. There are also some inflection points on the graph (where f’’(x)=0) and by making an initial guess that is too close to these points, Newtons method has the potential to converge to many different solutions, as I saw in my various initial guesses, which gave a point that was very different from that obtained by the golden search section method for an x0 that was too close to an inflection point as stated before. However, when I changed the tolerance to a larger value, Newton’s method would converge more quickly to the point that I obtained in the golden search section method. Before, with a smaller tolerance (1e-16), Newton’s method would step through the loop in far more steps and still not converge exactly to the minimum value.



```
%Problem 1 part b
f=@(x)(sin(tan(x))-tan(sin(x)));
tolerance=1e-16;
a=1.5646;
b=1.5647;
c=(-1+sqrt(5))/2;

x=c*a+(1-c)*b;
fx=f(x);
y=(1-c)*a+c*b;
fy=f(y);

for k=1:1000
    if fx<fy
        b=y;
        y=x;
        fy=fx;
        x=c*a+(1-c)*b;
        fx=f(x);
    else
        a=x;
        x=y;
        fx=fy;
        y=(1-c)*a+c*b;
        fy=f(y);
    end
    if (b-a) < tolerance
        break;
    end
end

xmin=b


%Using Newton's method
fprime=@(x)(cos(tan(x))*(sec(x).^2)-(sec(sin(x)).^2)*cos(x));
```

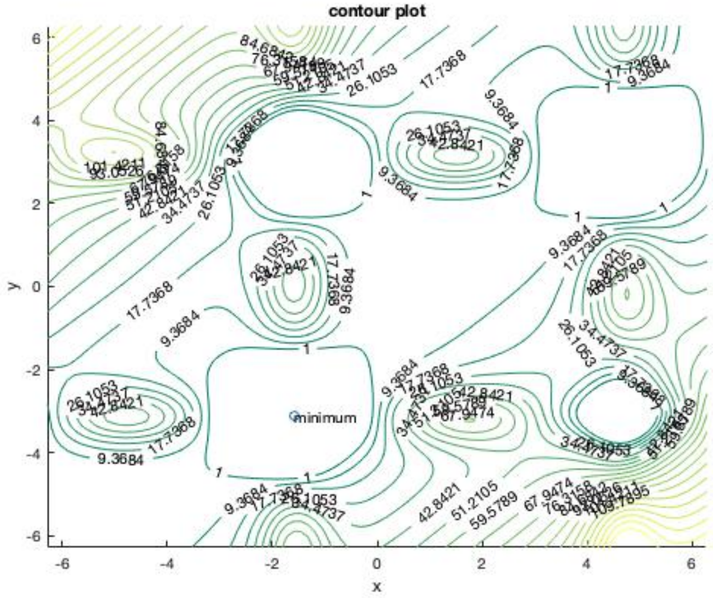
```
fdprime=@(x)((-sec(x).^4)*sin(tan(x))+2*(sec(x).^2)*tan(x)*cos(tan(x))-
2*sec(sin(x).^2)*(cos(x).^2)*tan(sin(x))+(sec(sin(x)).^2)*sin(x))

%initial guess x0=b from goldensection search method
x0=b;
tolerance=1e-16;
X=zeros(1,101);
X(1)=x0;

for i=1:100
    X(i+1)=X(i)-fprime(X(i)) / fdprime(X(i));
    if abs(fprime(X(i+1))) < tolerance
        break;
    end
end

xmin2=X(i+1)


%Intial guess x0=2
x0=2;
tolerance=1e-16;
X=zeros(1,101);
X(1)=x0;

for i=1:100
    X(i+1)=X(i)-fprime(X(i)) / fdprime(X(i));
    if abs(fprime(X(i+1))) < tolerance
        break;
    end
end

xmin3=X(i+1)


%Initial guess x0=1
x0=1;
tolerance=1e-16;
X=zeros(1,101);
X(1)=x0;

for i=1:100
    X(i+1)=X(i)-fprime(X(i)) / fdprime(X(i));
    if abs(fprime(X(i+1))) < tolerance
        break;
    end
end

xmin4=X(i+1)



%Initial guess x0=1.5
x0=1.5;
tolerance=1e-16;
X=zeros(1,101);
X(1)=x0;

for i=1:100
    X(i+1)=X(i)-fprime(X(i)) / fdprime(X(i));
    if abs(fprime(X(i+1))) < tolerance
        break;
    end
end

xmin5=X(i+1)


%Initial guess x0=1.6

x0=1.6;
tolerance=1e-16;
X=zeros(1,101);
X(1)=x0;

for i=1:100
    X(i+1)=X(i)-fprime(X(i)) / fdprime(X(i));
    if abs(fprime(X(i+1))) < tolerance
```

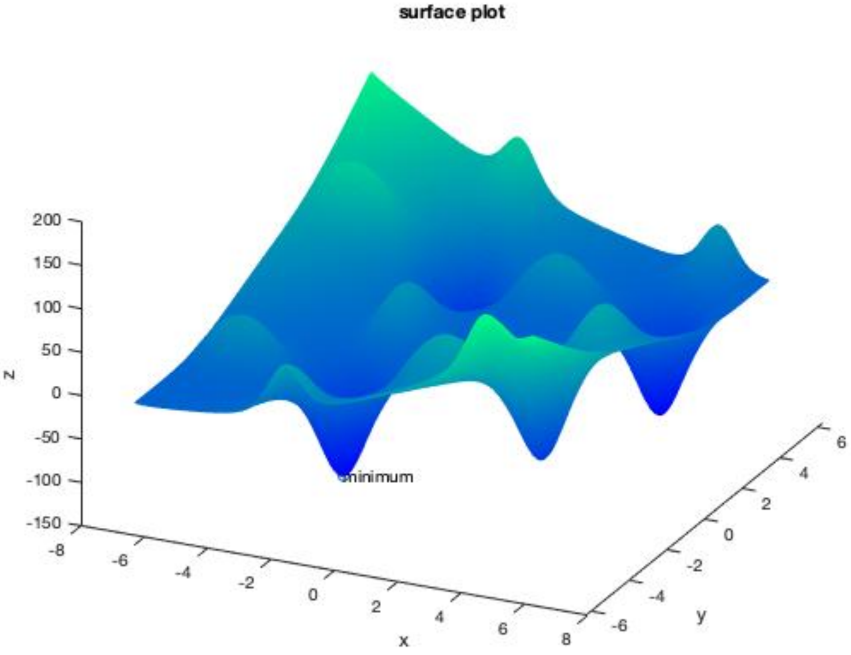
```
        break;
    end
end

xmin6=X(i+1)

%change of tolerance to larger value
x0=1.6;
tolerance=1e-7;
X=zeros(1,101);
X(1)=x0;

for i=1:100
    X(i+1)=X(i)-fprime(X(i)) / fdprime(X(i));
    if abs(fprime(X(i+1))) < tolerance
        break;
    end
end

xmin7=X(i+1)
```

c) For this function, $f'(x) = \frac{x-2}{2|x-2|^{\frac{1}{2}}}$ and $f''(x) = \frac{(x-2)^2}{2|x-2|^{\frac{5}{2}}}$
In using Newton’s method for this function, the minimum never converges. The vector X that I set to hold values for xmin go back and forth between 3 and 1. This is because at the minimum of x=2, f’(x) has an inflection points and we know that if the initial guess is too close to an inflection point, Newton’s method has little hope of converging.

```
%Problem 1 part C
f=@(x)(2/3*(abs(x-2))^(3/2));
fprime=@(x)((x-2)/((abs(x-2))^(1/2)));
fdprime=@(x)(((x-2)^2)/(2*abs(x-2)^(5/2)));

tolerance=1e-10;
X=zeros(1,1001);
x0=3;
X(1)=x0;

for k=1:1000
    X(k+1)=X(k)-fprime(X(k))/fdprime(X(k));
    if abs(fprime(X(k+1))) < tolerance
        break;
    end
end

X=X(1:k+1)
```

Problem 2


contour plot
y
x
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6
minimum
1
9.3684
17.7368
26.1053
34.4737
42.8421
51.2105
59.5789
67.9474
76.3158
84.6842
93.0526
101.4211
109.7895


surface plot
z
200
150
100
50
0
-50
-100
-150
x
-8
-6
-4
-2
0
2
4
6
8
y
-6
-4
-2
0
2
4
6
minimum

```
clear all; close all; clc
%Problem 1 part A

%Section Search and golden search for A_114 with a=1, b=2, c=0.5001
v1=zeros(1,114)
v1(1,:)=2;
A=diag(v1);
v2=zeros(1,113);
v2(1,:)=-1;
B=diag(v2,1);
C=diag(v2,-1);
A_114=A+B+C;

max_abs_eig_A114=@(w)(f(w, diag(diag(A_114)), tril(A_114,-1), triu(A_114,+1)))
tic
w = section_search(max_abs_eig_A114, 1, 2, 0.5001)
toc

tic
w2 = golden_section_search(max_abs_eig_A114, 1, 2)
toc


%Section search and golden search methods of A_1000
v12=zeros(1,1000)
v12(1,:)=2;
A2=diag(v12);
v22=zeros(1,999);
v22(1,:)=-1;
B2=diag(v22,1);
C2=diag(v22,-1);
A_1000=A2+B2+C2;


max_abs_eig_A1000 = @(w)(f(w, diag(diag(A_1000)), tril(A_1000, -1), triu(A_1000, +1)));
tic
w_1000 = section_search(max_abs_eig_A1000, 1, 2, 0.5001)
toc

tic
w2_1000 = golden_section_search(max_abs_eig_A1000, 1, 2)
toc
function y=f(w,D,L,U)
    P=1/w*D+L;
    T=(w-1)/w*D+U;
    M=-P\T;
    y=max(abs(eig(M)));
end
function x = section_search(f, a, b, c)
    tolerance = 1e-8;
    for k=1:1000
        x = c*a + (1-c)*b;
        y = (1-c)*a + c*b;
        if f(x) < f(y)
            b = y;
        else
            a = x;
        end
        if (b-a) < tolerance
            break;
        end
    end
end
function x = golden_section_search(f,a,b)
    c = (-1 + sqrt(5)) / 2;
    tolerance = 1e-8;
    x = c*a + (1-c)*b;
    fx = f(x);
    y = (1-c)*a + c*b;
    fy = f(y);
    for k = 1:1000
        if fx < fy
            b = y;
            y = x;
            fy = fx;
            x = c*a + (1-c)*b;
            fx = f(x);
        else
            a = x;
            x = y;
            fx = fy;
            y = (1-c)*a +c*b;
            fy = f(y);
        end
        if (b-a) < tolerance
            break;
        end
    end
end
```