Technion – Israel Institute of Technology

HW1

Numerical Methods

019003

Alon Spinner	305184335	alonspinner@gmail.com
Oren Elmakis	311265516	orenelmakis@gmail.com

October 31, 2021

Question 1

a.

Here below we wrote the function my “MyDist_a” which computes the Euclidean distance between a set of points P and an origin points P_0

```
function R=MyDist_a(P,P0)
R=sqrt(sum((P-P0).^2,2));
end
```

b.

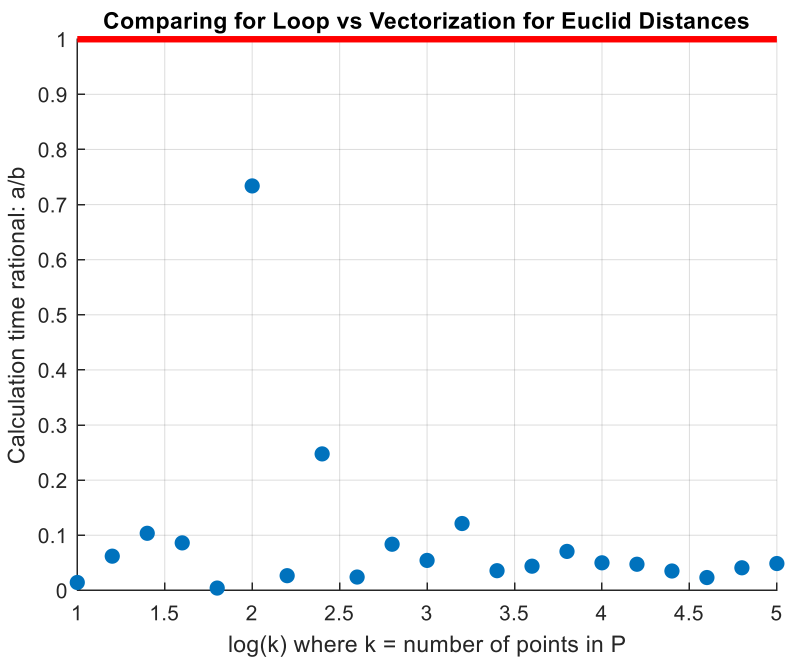
The function myDist_b wastes time and resources by calling MyDist_a iteratively for each points in P

```
function R=MyDist_b(P,P0)
    R=zeros(size(P,1),1);
    for ii=1:length(R)
        R(ii)=MyDist_a(P(ii),P0);
    end
end
```

c.

In this question we compared the two different methods by computing the time it takes for the two functions to run on the same sets of points.
We ran the test for different number of points in the range of $10 \div 10^5$

```
P0=[0.5,0.5];
logk=1:0.2:5;
k=10.^logk; %amount of points in Pk

[a,b]=deal(zeros(size(k)));
for ii=1:length(k)
    P=rand(round(k(ii)),2);
    tic;
    R=MyDist_a(P,P0);
    a(ii)=toc;

    tic;
    R=MyDist_b(P,P0);
    b(ii)=toc;
end
```



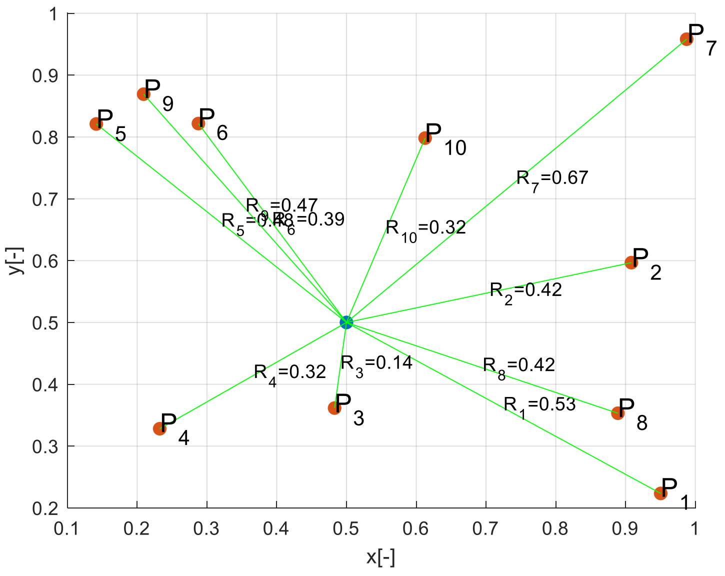
As you can clearly see, the ratio time ratio $\frac{a}{b}$ is way below 1. In fact, it seems that method a is more than 10 times faster than method b.

d.

With some magic tricks, we were able to render the figure below, displaying random points and their distances to some origin point

The text is displayed in a location by the following formula:

$$Q_{ii} = \frac{P_{ii} - P_0}{2} + P_0$$



Q2.

a.

The analytic Taylor polynomials are calculated based on a recursive formula for general 2D derivatives of the taylor series.

```
function poly = Taylor_fun(n,f,point_x, point_y)
    syms x y
    poly = subs(subs(f,x,point_x),y,point_y);
    for i = 1:n
        poly = poly + Taylor_part_fun(1,i,f,point_x, point_y);
    end
end


function [poly] = Taylor_part_fun(j,n,f,point_x,point_y)
    syms x y
    if j > n
        poly = subs(subs(f,x,point_x),y,point_y);
    else
        poly = Taylor_part_fun(j+1,n,diff(f,x),point_x,point_y)/j*(x-
point_x)+Taylor_part_fun(j+1,n,diff(f,y),point_x,point_y)/j*(y-
point_y);
    end
end
```

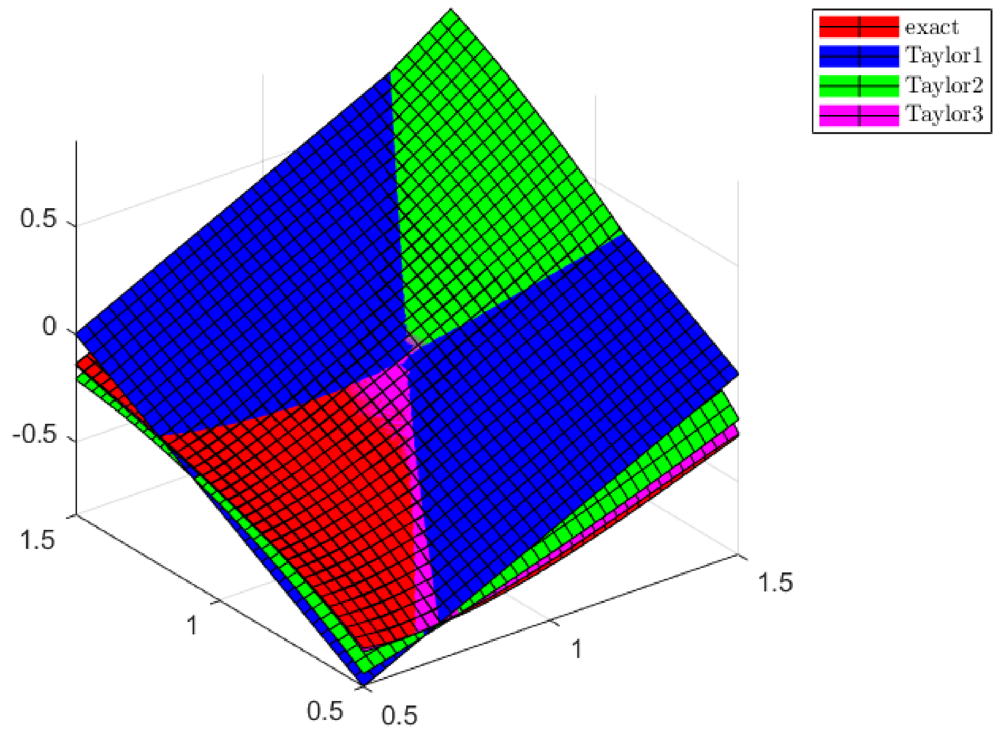
$$poly_1 = sin(1)*(x-1) + sin(1)*(y-1)$$

$$poly_2 = sin(1)*(x-1) + sin(1)*(y-1) + ((cos(1)*(x-1))/2 - (sin(1)*(y-1))/2)*(y-1) + ((cos(1)*(y-1))/2 + (cos(1) - sin(1)/2)*(x-1))*(x-1)$$

$$poly_3 = sin(1)*(x-1) - (((sin(1)*(y-1))/6 + ((cos(1) + sin(1)/3)*(x-1))/2)*(x-1) + ((cos(1)*(y-1))/6 + (sin(1)*(x-1))/6)*(y-1))*(x-1) - (((cos(1)*(y-1))/6 + (sin(1)*(x-1))/6)*(x-1) + ((cos(1)*(x-1))/6 - (sin(1)*(y-1))/3)*(y-1))*(y-1) + sin(1)*(y-1) + ((cos(1)*(x-1))/2 - (sin(1)*(y-1))/2)*(y-1) + ((cos(1)*(y-1))/2 + (cos(1) - sin(1)/2)*(x-1))*(x-1)$$

b.

we present below the output graph for plane fitting. we did not compute RMS to confirm our results numerically.



The magenta surface which represents Taylor approximation of the third degree is “the closest” to the red surface.

Q3

We heavily edited the code provided to us, which aimed to solve the two equations using Newton’s method:

$$x^2 + y^2 = 4$$

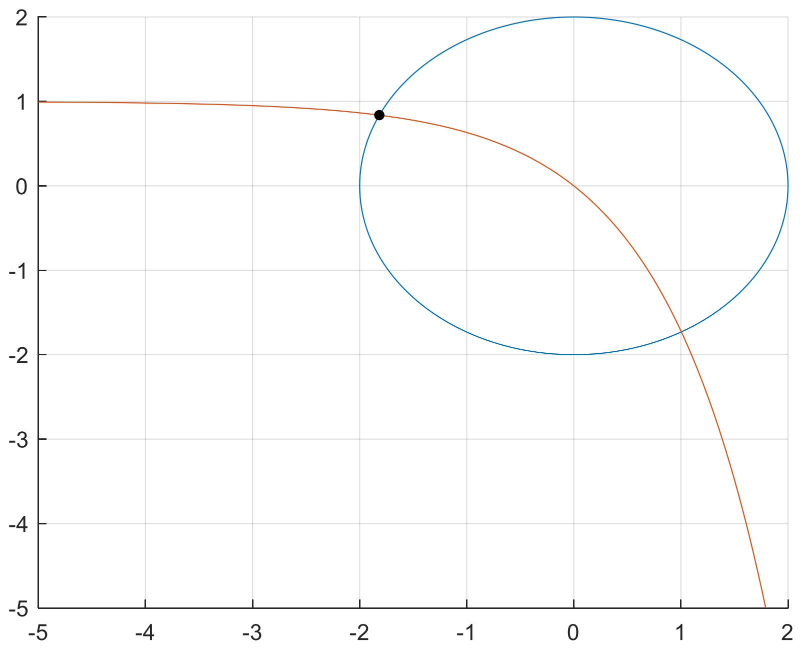
$$e^2 + y = 1$$

```
e=1e-2; %FIX: changed from 1e5
xkm1=[1; 1];
xk=xkm1+2*e; %just to make sure we enter the first loop iteration
iterN=0; %fixed naming convention. Start with lower letters

fnF=@(x) [x(1)^2+x(2)^2-4 ;
        exp(x(1))+x(2)-1]; %fix from old code: turned + const to -
fnJ=@(x) [2*x(1) 2*x(2);
        exp(x(1)) 1];

while norm(xk-xkm1,2)>e %Euclid norm
    xkm1=xk;
    F=fnF(xkm1);
    J=fnJ(xkm1);
    dx=-J\F; %FIX: changed from right divide to left divide. we need
to solve J*dx=-F
    xk=xkm1+dx;
    iterN=iterN+1;
end
```

Printing our solution to the given initial condition provided the following:

```
with interations = 6
xk=-1.81626,0.837368
F(xk)=0.00494268,0.000334923
```

We also ploted the solution to confirm



As is shown, our solution (black) converged on one of the two solutions (intersections between two graphs)

Q4

The following pseudocode describe the flow-chart

```
INPUT X
IF X<10 THEN
    IF X<5 THEN
        X = 5
    ELSE
        PRINT X
    END
ELSE
    DO
        IF X<50 EXIT
        ELSE
            X = X-5
    ENDDO
ENDIF
```