Application Problems:

Problem 2.6: First-order Approximation of functions

a)

```
% Create Syms representation of variables for differentiation
syms x y

% Functions
f1(x,y) = 2.*x+3.*y+1
```

f1(x, y) = $2x + 3y + 1$

```
f2(x,y) = x.^2 + y.^2 - x.*y -5
```

f2(x, y) = $x^2 - x\,y + y^2 - 5$

```
f3(x,y) = (x-5).*cos(y-5) - (y-5).*sin(x-5)
```

f3(x, y) = $\cos(y-5)\,(x-5) - \sin(x-5)\,(y-5)$

f1(x, y) = $2x + 3y + 1$

f2(x, y) = $x^2 - x\,y + y^2 - 5$

f3(x, y) = $\cos(y-5)\,(x-5) - \sin(x-5)\,(y-5)$


```
% Compute gradients
grad_f1 = gradient(f1,[x,y])
```

grad_f1(x, y) =

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

```
grad_f2 = gradient(f2,[x,y])
```

grad_f2(x, y) =

$$\begin{pmatrix} 2x - y \\ 2y - x \end{pmatrix}$$

```
grad_f3 = gradient(f3,[x,y])
```

grad_f3(x, y) =

$$\begin{pmatrix} \cos(y-5) - \cos(x-5)\,(y-5) \\ -\sin(x-5) - \sin(y-5)\,(x-5) \end{pmatrix}$$

grad_f1(x, y) =

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

grad_f2(x, y) =

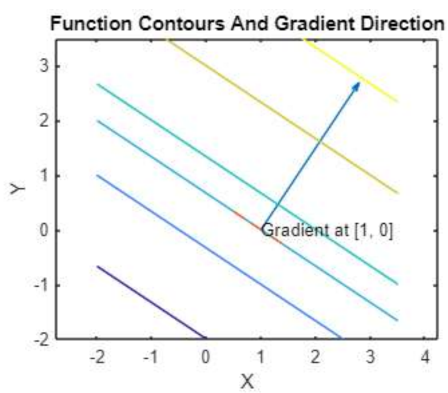
$$\begin{pmatrix} 2x - y \\ 2y - x \end{pmatrix}$$

grad_f3(x, y) =

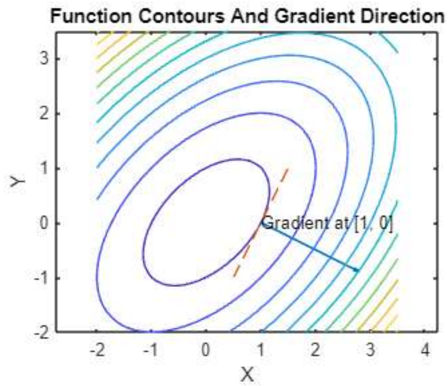
$$\begin{pmatrix} \cos(y-5) - \cos(x-5)\,(y-5) \\ -\sin(x-5) - \sin(y-5)\,(x-5) \end{pmatrix}$$

b)

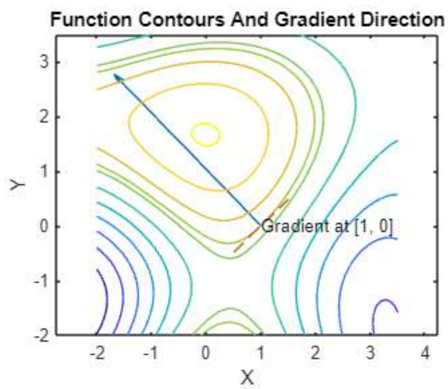
```
contourPlot(f1, grad_f1, [1,0])
```

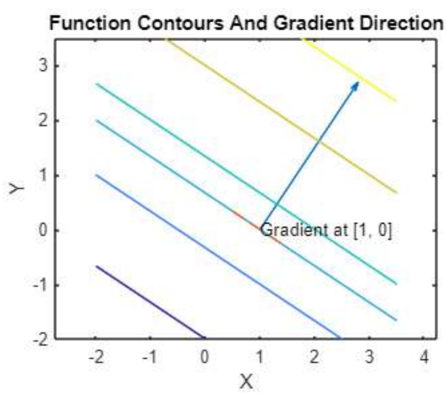


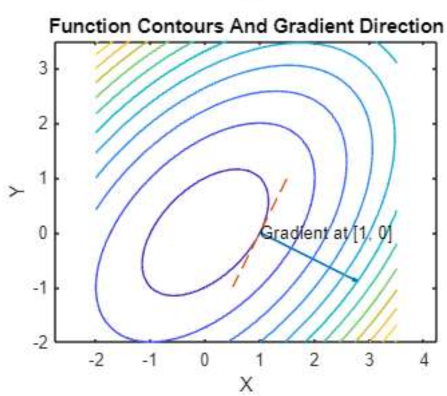
```
contourPlot(f2, grad_f2, [1,0])
```

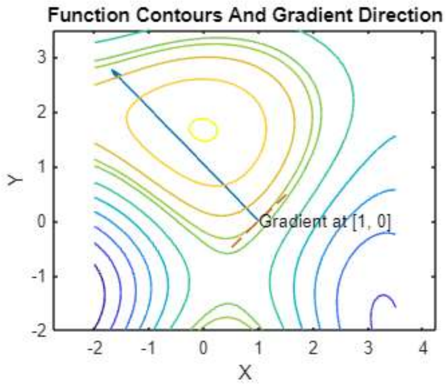



```
contourPlot(f3, grad_f3, [1,0])
```



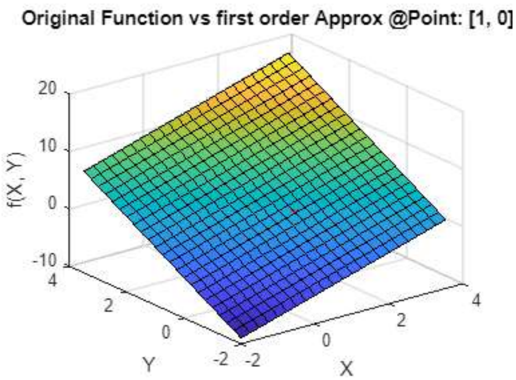




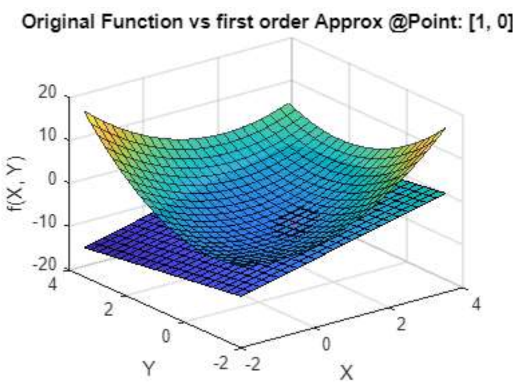


c)

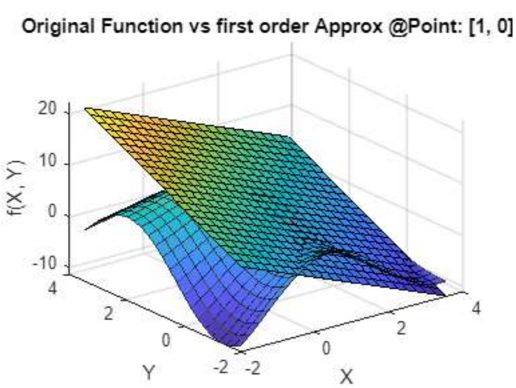
```
taylorMapp(f1, 2, [1 0])
```

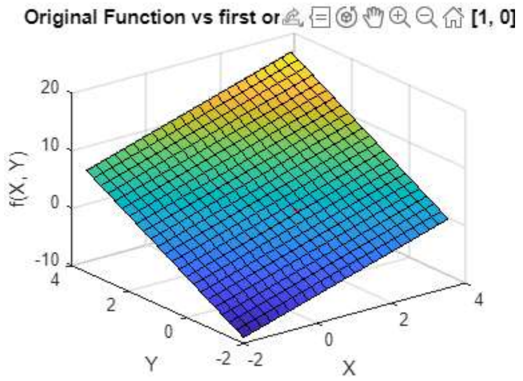


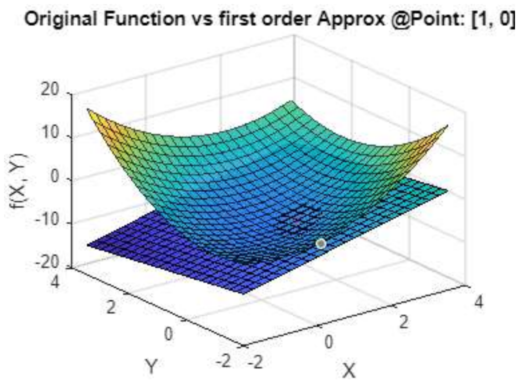

```
% It should be noted that the original function is a plane
% So the approximation is also just a plane
taylorMapp(f2, 2, [1 0])
```

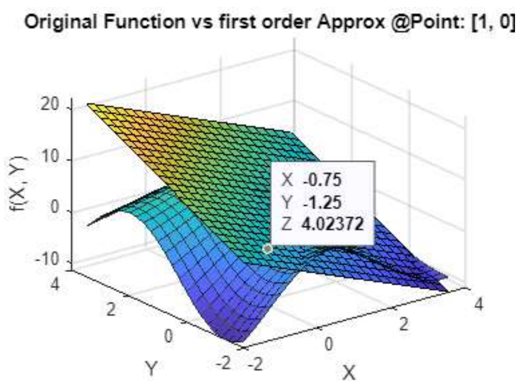



```
taylorMapp(f3, 2, [1 0])
```









Problem 2.7: Second-order approximation of functions

a)

```
% Hessian
hes_f1 = hessian(f1, [x,y])
```

hes_f1(x, y) =

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

```
hes_f2 = hessian(f2, [x,y])
```

hes_f2(x, y) =

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

```
hes_f3 = hessian(f3, [x,y])
```

hes_f3(x, y) =

$$\begin{pmatrix} \sin(x-5)\ (y-5) & -\cos(x-5) - \sin(y-5) \\ -\cos(x-5) - \sin(y-5) & -\cos(y-5)\ (x-5) \end{pmatrix}$$

```
hes_f1(x, y) =
```

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

```
hes_f2(x, y) =
```

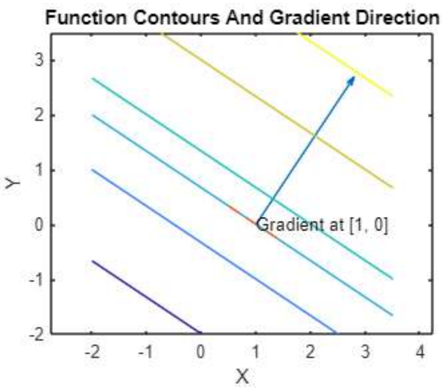
$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

```
hes_f3(x, y) =
```

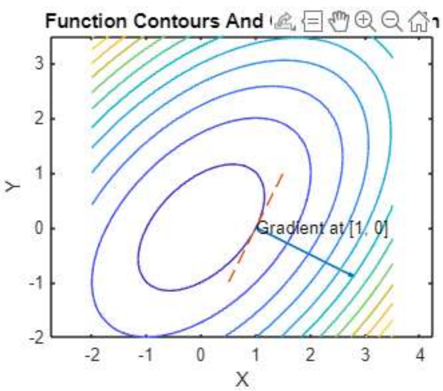
$$\begin{pmatrix} \sin(x-5)\,(y-5) & -\cos(x-5)-\sin(y-5) \\ -\cos(x-5)-\sin(y-5) & -\cos(y-5)\,(x-5) \end{pmatrix}$$

b)

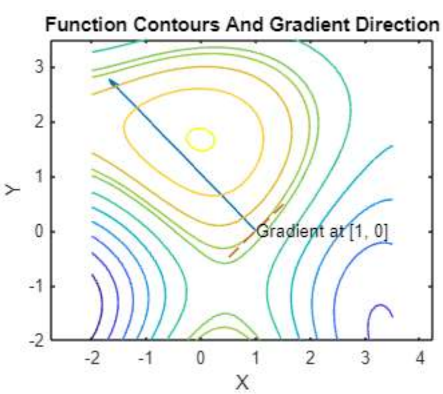
```
% 2D Contour plot from previous question
contourPlot(f1, grad_f1, [1,0])
```

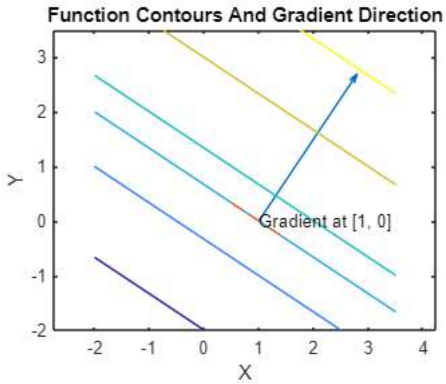


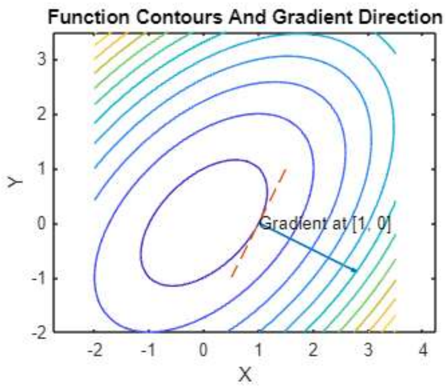

```
contourPlot(f2, grad_f2, [1,0])
```

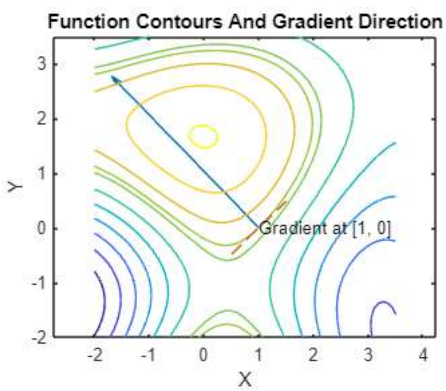



```
contourPlot(f3, grad_f3, [1,0])
```

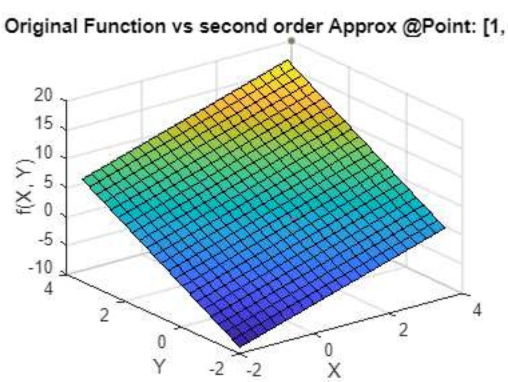




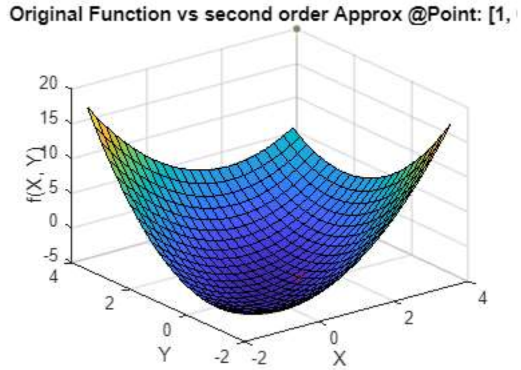




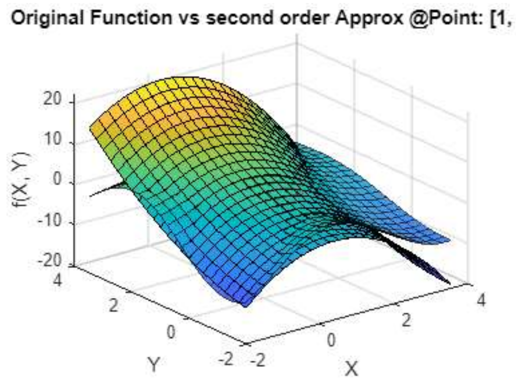

```
% 3D plots
[xx, yy] = meshgrid(-2:0.25:3.5);
% Same function as last time, just now a higher order
taylorMapp(f1, 3, [1 0]) % Linear so the approx is exact
```

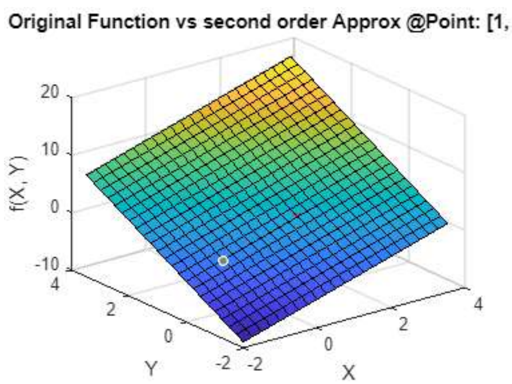


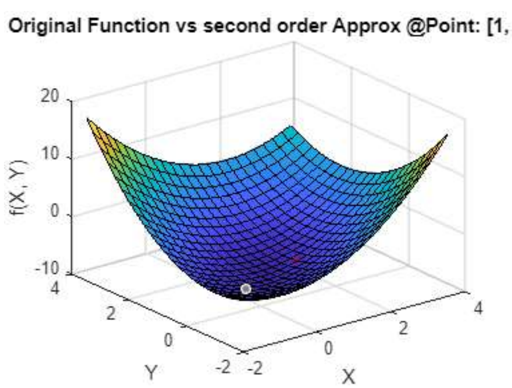

```
taylorMapp(f2, 3, [1 0]) % Quadratic so this time the approx is exact
```

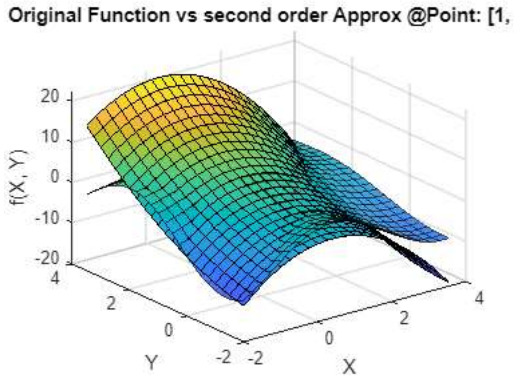



```
taylorMapp(f3, 3, [1 0])
```





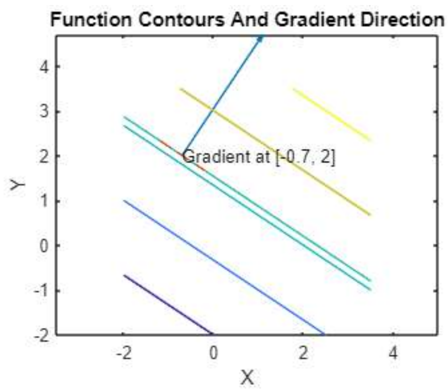




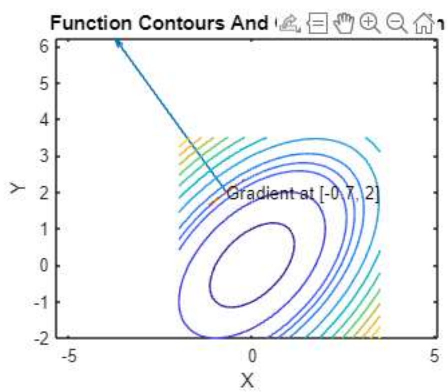
c)

For point (-0.7,2)

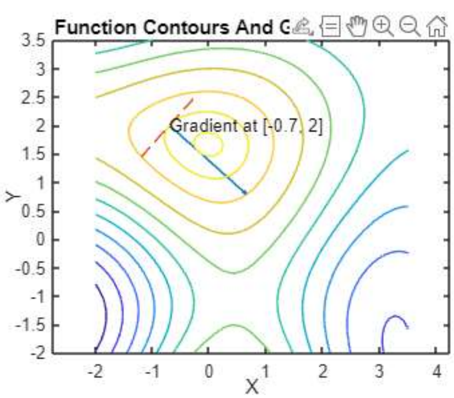
```
contourPlot(f1, grad_f1, [-0.7 2])
```



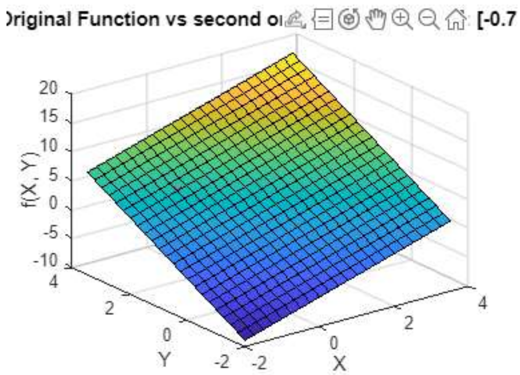

```
contourPlot(f2, grad_f2, [-0.7 2])
```



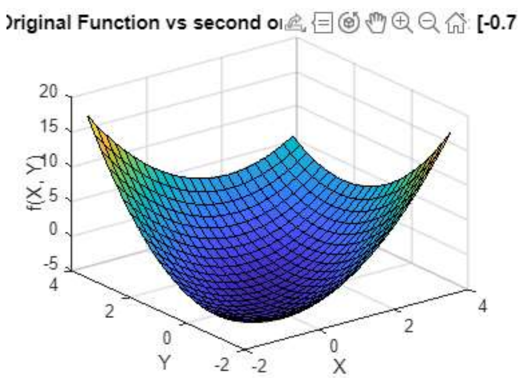

```
contourPlot(f3, grad_f3, [-0.7 2])
```



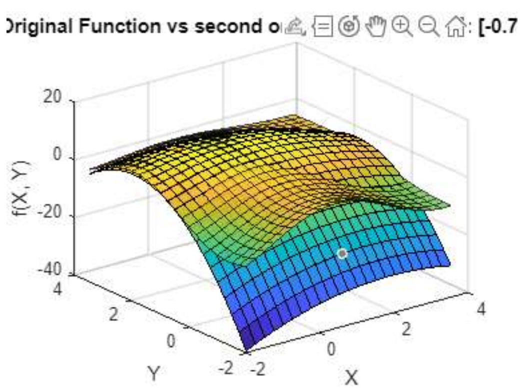

```
taylorMapp(f1, 3, [-0.7 2]) % Linear so the approx is exact
```

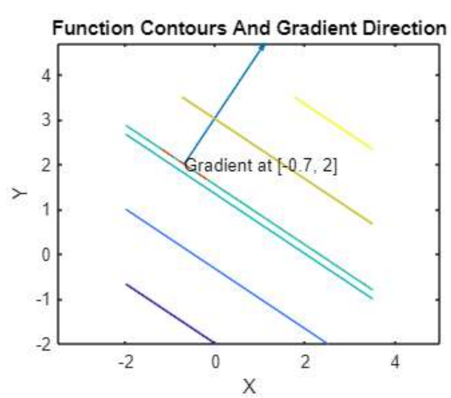


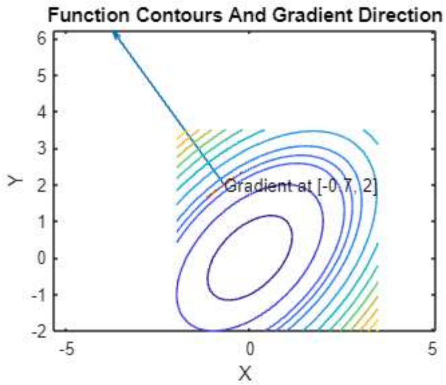

```
taylorMapp(f2, 3, [-0.7 2]) % Quadratic so this time the approx is exact
```

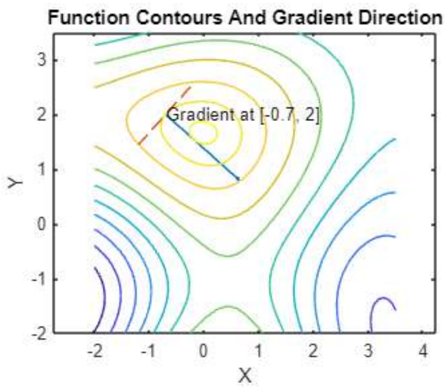


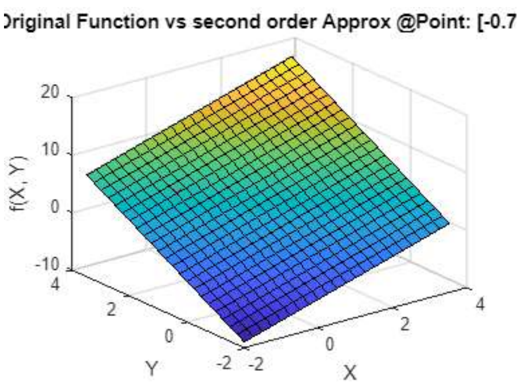

```
taylorMapp(f3, 3, [-0.7 2])
```

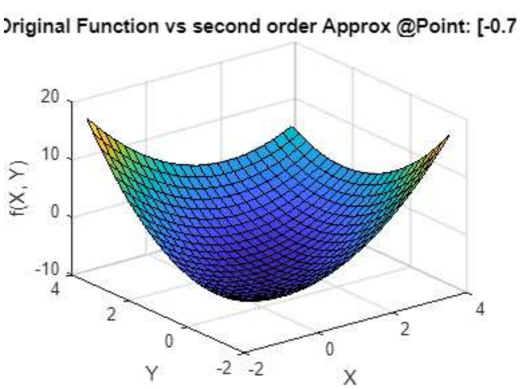


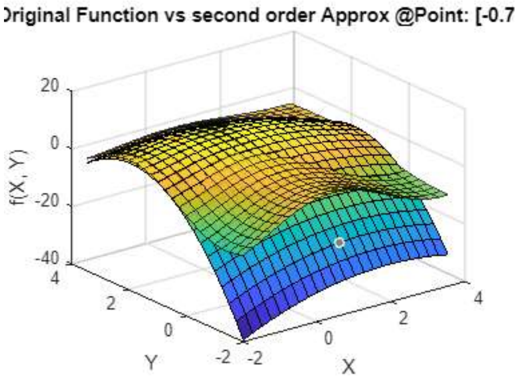



Function Contours And Gradient Direction
Gradient at [-0.7, 2]
X
Y


Function Contours And Gradient Direction
Gradient at [-0.7, 2]
X
Y

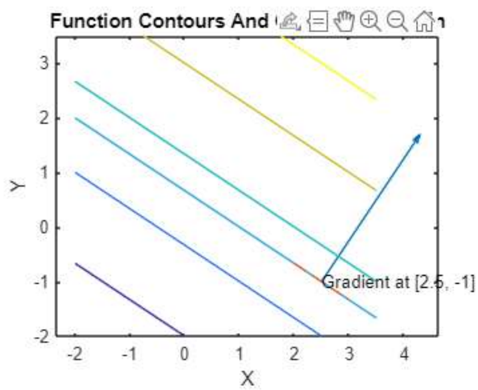

)riginal Function vs second order Approx @Point: [-0.7
f(X, Y)
X
Y


)riginal Function vs second order Approx @Point: [-0.7
f(X, Y)
X
Y

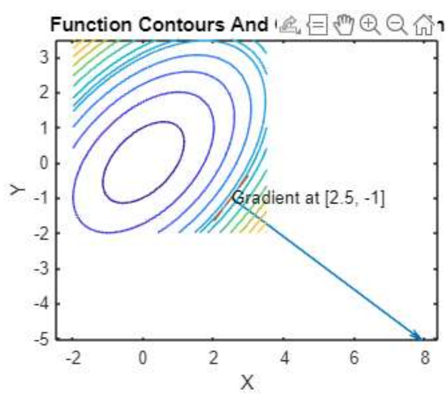


For point (2.5, -1)

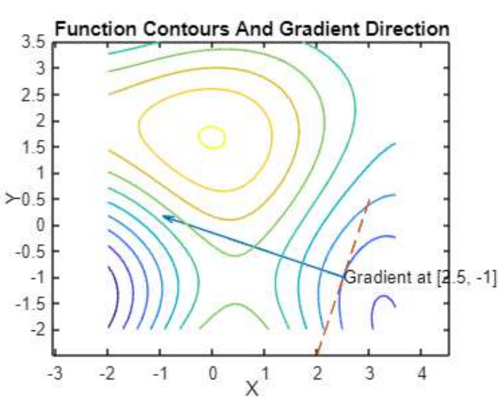
```
contourPlot(f1, grad_f1, [2.5 -1])
```



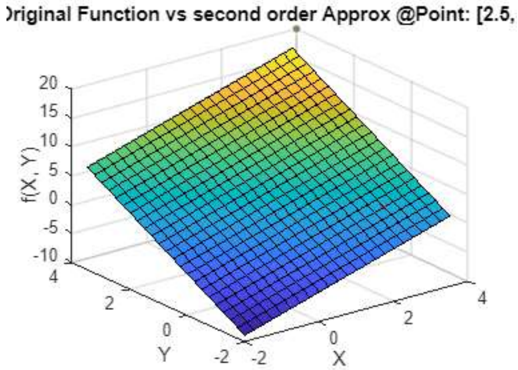

```
contourPlot(f2, grad_f2, [2.5 -1])
```



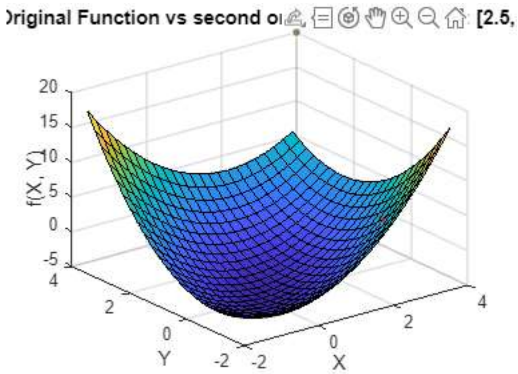

```
contourPlot(f3, grad_f3, [2.5 -1])
```



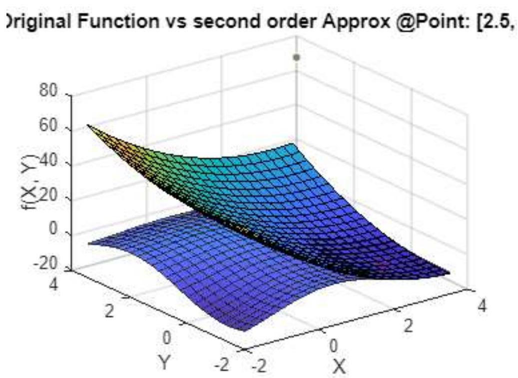

```
taylorMapp(f1, 3, [2.5 -1]) % Linear so the approx is exact
```

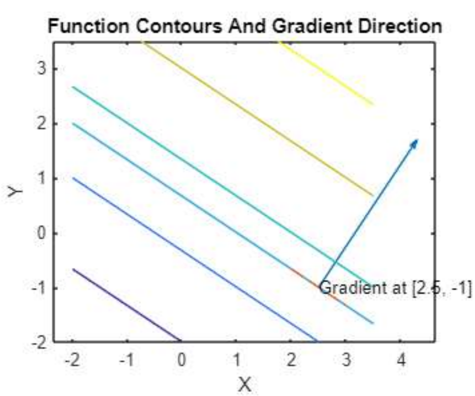


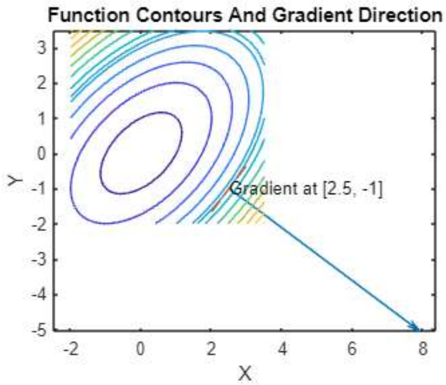

```
taylorMapp(f2, 3, [2.5 -1]) % Quadratic so this time the approx is exact
```

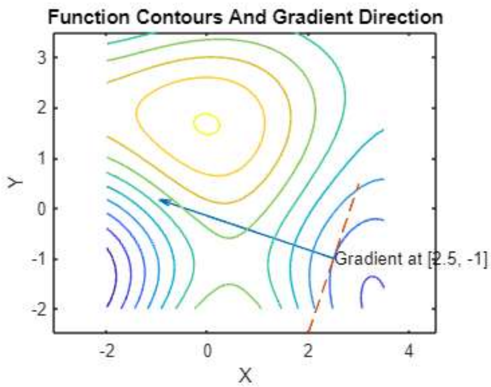


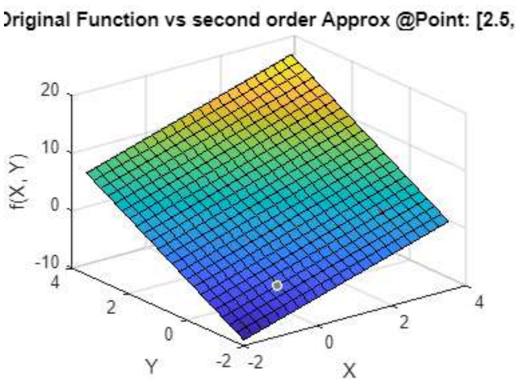

```
taylorMapp(f3, 3, [2.5 -1])
```

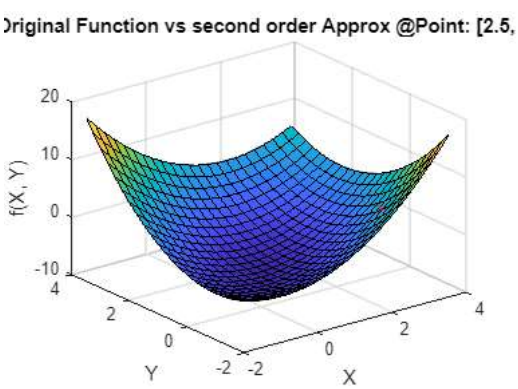


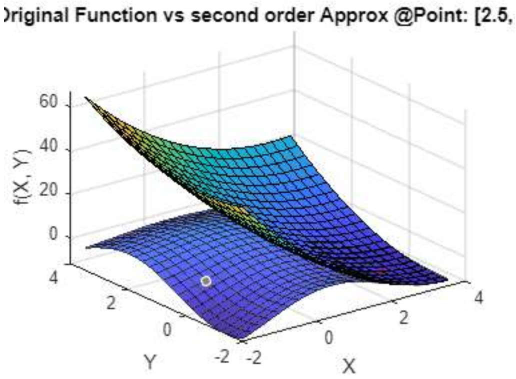



Function Contours And Gradient Direction
Gradient at [2.5, -1]
X
Y


Function Contours And Gradient Direction
Gradient at [2.5, -1]
X
Y


)riginal Function vs second order Approx @Point: [2.5,
f(X, Y)
X
Y


)riginal Function vs second order Approx @Point: [2.5,
f(X, Y)
X
Y



d)

The 1st and 2nd order approximations are accureate for the 1st and 2nd order functions. For the sinusoid however, neither approximation was accurate. There were issues on each graph. The reason that the other two approximations were accurate was because the closest 2nd order approixmation of a 2nd order function would of course be the actual function. Same goes for the first order approximation and function. The sinusoidal function however cannot be accurately represented at such a low order. Only when the function is close to the point of interest is the approximation accurate, which can be useful. However this approximation cannot model the whole function at once.

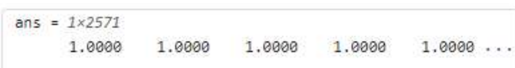
Problem 2.8: Google's PageRank algorithm

a)

```
load 'pagerank_adj.mat'
```

```
A = J ./ sum(J);
sum(A) % sum all columns
```

```
ans = 1×2571
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

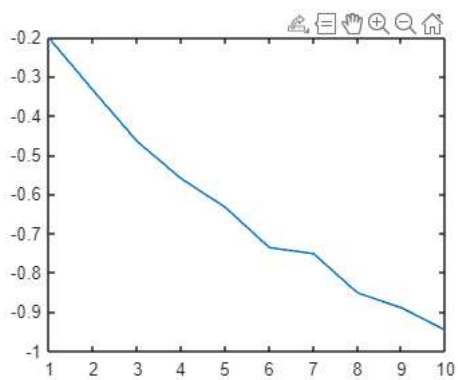


```
ans = 1×2571
    1.0000    1.0000    1.0000    1.0000    1.0000 ...
```

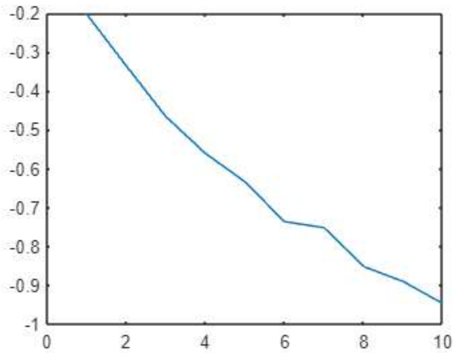
All the columns summed equals one. This is because of the law of probability, which states that all probabilities must sum to 1, and the fact that the model doesn't let the state/web page feed back into itself. Thus all the entries in each column will sum up to zero.

b)

```
k = 10;
e1 = powerIteration(A,k);
```



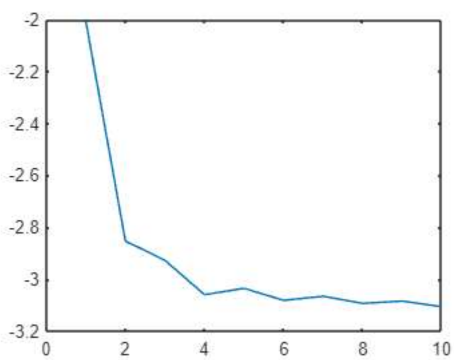
```
WorstScore = 1×5
     0     0     0     0     0
WorstPage = 1×5
     1     3     4     5    10
BestScore = 1×5
    0.4394    0.3296    0.3073    0.2996    0.2751
BestPage = 1×5
     2    35    36    58    49
```



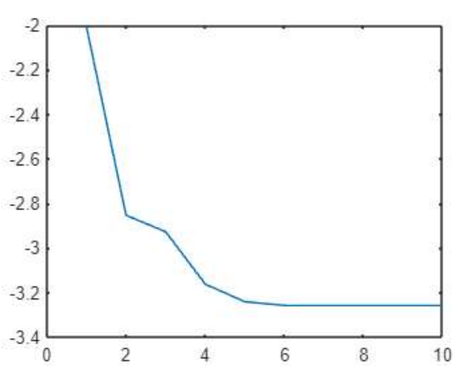

```
WorstScore = 1×5
     0     0     0     0     0

WorstPage = 1×5
     1     3     4     5    10

BestScore = 1×5
    0.4394    0.3296    0.3073    0.2996    0.2751

BestPage = 1×5
     2    35    36    58    49
```

c)

```
e2 = shiftInvertPI(A,k,0.99); % sigma value supplied in question
```



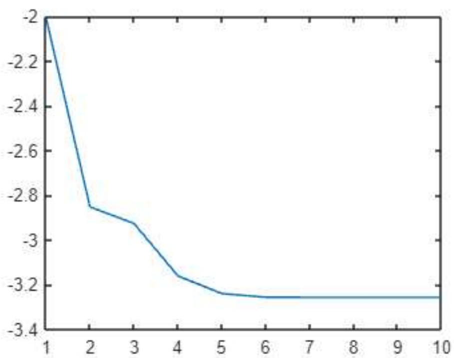

```
WorstScore = 1×5
   -0.3196   -0.1602   -0.1602   -0.1200   -0.0917
WorstPage = 1×5
   424   987   986   985   930
BestScore = 1×5
    0.3711    0.3184    0.2975    0.2904    0.2637
BestPage = 1×5
     2    35    36    58    49
```



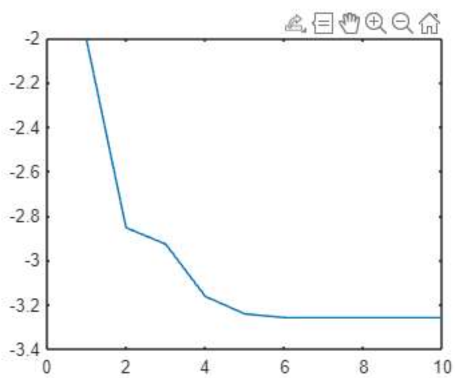
```
WorstScore = 1×5
    -0.3196   -0.1602   -0.1602   -0.1200   -0.0917

WorstPage = 1×5
    424   987   986   985   930

BestScore = 1×5
     0.3711    0.3184    0.2975    0.2904    0.2637

BestPage = 1×5
      2    35    36    58    49
```

```
e3 = rayleighQuotientI(A,k,0.99);
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.192291e-16.
```

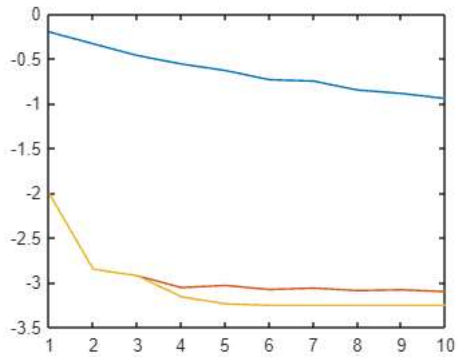


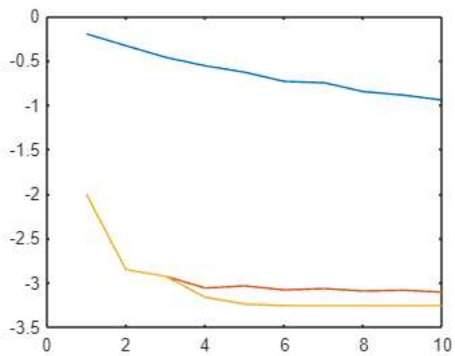

```
WorstScore = 1×5
   -0.5458   -0.2732   -0.2732   -0.2048   -0.1536
WorstPage = 1×5
   424   987   986   985   930
BestScore = 1×5
    0.2576    0.2211    0.2065    0.2016    0.1831
BestPage = 1×5
     2    35    36    58    49
```




```
WorstScore = 1×5
    -0.5458   -0.2732   -0.2732   -0.2048   -0.1536

WorstPage = 1×5
    424   987   986   985   930

BestScore = 1×5
     0.2576    0.2211    0.2065    0.2016    0.1831

BestPage = 1×5
      2    35    36    58    49
```


```
plot(1:k,log10(e1),1:k,log10(e2),1:k,log10(e3))
```





d)

Top Pages: (All 3 Algorithms agree)

http://www.hollins.edu/ (index 2)

http://www.hollins.edu/admissions/visit/visit.htm (index 35)

http://www.hollins.edu/about/about_tour.htm (index 36)

http://www.hollins.edu/htdig/index.html (index 58)

http://www.hollins.edu/admissions/info-request/info-request.cfm (index 49)

Inuitively it makes sense that the top page would be the main hollins webpage as all the websites would have a button to go back to it since its the main page. The next ones being about admissions and tours also makes sense since entering the school would be the next biggest reason to visit the website

Bottom Pages: (Only ShiftInvertPI and RayleighQuotient agrees)

http://www1.hollins.edu/homepages/hammerpw/qrhomepage.htm (Index 424)

http://www1.hollins.edu/homepages/hammerpw/qrcourses2.htm (Index 987)

http://www1.hollins.edu/homepages/hammerpw/qrcourses.htm (index 986)

http://www1.hollins.edu/homepages/hammerpw/qractivities.htm (Index 985)

http://www1.hollins.edu/homepages/godardrd/homepage.htm (Inde 930)

Most of these pages seem to be centered around this hammerpw thing which I couldn't figure out what it was, but it seems like they are all sources from qr codes, which means they wouldn't be normally accessible through the system, only through the QR codes. So intuituivly this result also makes sense.

Helper Functions:

```
function contourPlot(f, F, position) % Plot a contour plot with Gradient
    % Initialize Variables
    syms x y;

    %  Use taylor series to get gradient
    approx = taylor(f,[x,y],position,'Order',2);
    apx(x) = solve(approx,y);
    tangent(x) = solve(approx,y) + position(2) - apx(position(1));

    % Solve for direction of Gradient
    uv = double(F(position(1),position(2)));

    % Plot Gradient
    quiver(position(1), position(2), uv(1),uv(2));
    hold on

    % Plot Contour 1
    fcontour(f, [-2 3.5]) % Specifies the range of values
```

```
    % Plot remaning Contours
    fcontour(f,[-2 3.5],'LevelList',[double(f(position(1), position(2)))])
    plot([position(1)-0.5; position(1)+0.5], [double(tangent(position(1)-0.5));double(tangent(position(1)+0.5))],'--') % Line Orthogonal w

    % Labeling
    text(position(1),position(2),"Gradient at [" + string(position(1)) + ", " + string(position(2)) + "]")
    axis equal
    xlabel('X');
    ylabel('Y');
    title('Function Contours And Gradient Direction');
    hold off
end

function taylorMapp(f, order, position) % Plot a 3-D Linear approximatio nof the function
    % Initialize Functions
    syms x y

    % Initialize meshgrid for surf function later
    [xx,yy] = meshgrid(-2:0.25:3.5);

    % Use Taylor series to get gradient
    approx = taylor(f,[x,y],position,'Order',order);
    figure; % Create a figure

    % Plot 3D Linear Approximation
    surf(xx,yy,double(f(xx,yy)),'EdgeAlpha',0.7,'FaceAlpha',0.9)
    hold on

    % Plot Tangent shape based on Taylor series
    surf(xx,yy,double(approx(xx,yy)))

    % Plot Orthogonal Vector to positon
    plot3(position(1),position(2),double(f(position(1),position(2))),'r*')

    % Labels
    xlabel('X');
    ylabel('Y');
    zlabel('f(X, Y)');
    if order == 2
        string_order = "first";
    end
    if order == 3
        string_order = "second";
    end
    title("Original Function vs " + string_order + " order Approx @Point: [" + string(position(1)) + ", " + string(position(2)) + "]");
    hold off
end
function error = powerIteration(A,k) %find eignevalues through repetition
    % Set some variables
    [N,~] = size(A);
    x = zeros(N,k+1);
    x(:,1) = ones(N,1); % Set first values of x to be 1

    % Set error values
    error = zeros(1,k); % Error is the lamda/eigenvalue in the text book
    error(1,1) = norm(A * x(:,1) - x(:,1)); % Set initial error value, based on the algorithm

    % Perform repeated operation
    for i = 1:k
        y = A * x(:,i);
        x(:,i+1) = y / norm(y);
        error(:,i) = norm(A * x(:,i+1) - x(:,i+1));
    end

    % Plot information
    figure;
    plot(1:k,log10(error)) % Plot log graph

    % Find the best and worst page
    [B,I] = sort(x(:,k+1));
    WorstScore = B(1:5)' % Because sorted
    WorstPage = I(1:5)'
    B = flip(B);
    I = flip(I);
    BestScore = B(1:5)' % Because sorted
    BestPage = I(1:5)'
end
function error = shiftInvertPI(A,k,sigma) % find eignevalue by guessing a sigma
    % Set some variables
    [N,~] = size(A);
```

```
    x = zeros(N,k+1);
    x(:,1) = ones(N,1);

    % Error = eignevalue
    error = zeros(1,k);

    % Use sigma to contruct a new , similar to A matrix
    newA = (A - sigma*eye(N));

    % Perform repeated operation
    for i = 1:k
        y = newA \ x(:,i);
        x(:,i+1) = y / norm(y);
        error(:,i) = norm(A * x(:,i+1) - x(:,i+1));
    end

    % Plot
    figure;
    plot(1:k,log10(error))

    % Find best and worst page
    [B,I] = sort(x(:,k+1));
    WorstScore = B(1:5)'
    WorstPage = I(1:5)'
    B = flip(B);
    I = flip(I);
    BestScore = B(1:5)'
    BestPage = I(1:5)'
end
function error = rayleighQuotientI(A,k,sigma) % Shift invert power but a more educated guess of sigma
    % Set some variables
    [N,~] = size(A);
    x = zeros(N,k+1);
    x(:,1) = ones(N,1);

    % Error is the Eigenvalue
    error = zeros(1,k);
    newA = (A - sigma*eye(N)); % New matrix based on sigma

    for i = 1:3
        y = newA \ x(:,i);
        x(:,i+1) = y / norm(y);
        error(:,i) = norm(A * x(:,i+1) - x(:,i+1));
    end

    % Do repeated parts in algorithm
    for i = 4:k
        sigmaK = (x(:,i)' * A * x(:,i))/( x(:,i)' * x(:,i) );
        y = (A - sigmaK*eye(N)) \ x(:,i);
        x(:,i+1) = y / norm(y);
        error(:,i) = norm(A * x(:,i+1) - x(:,i+1));
    end

    % Plot
    figure;
    plot(1:k,log10(error))

    % Find best and worst page
    [B,I] = sort(x(:,k+1));
    WorstScore = B(1:5)'
    WorstPage = I(1:5)'
    B = flip(B);
    I = flip(I);
    BestScore = B(1:5)'
    BestPage = I(1:5)'
end
```