Exam 3

Math 3607, Summer 2021

[SMITH.11783]

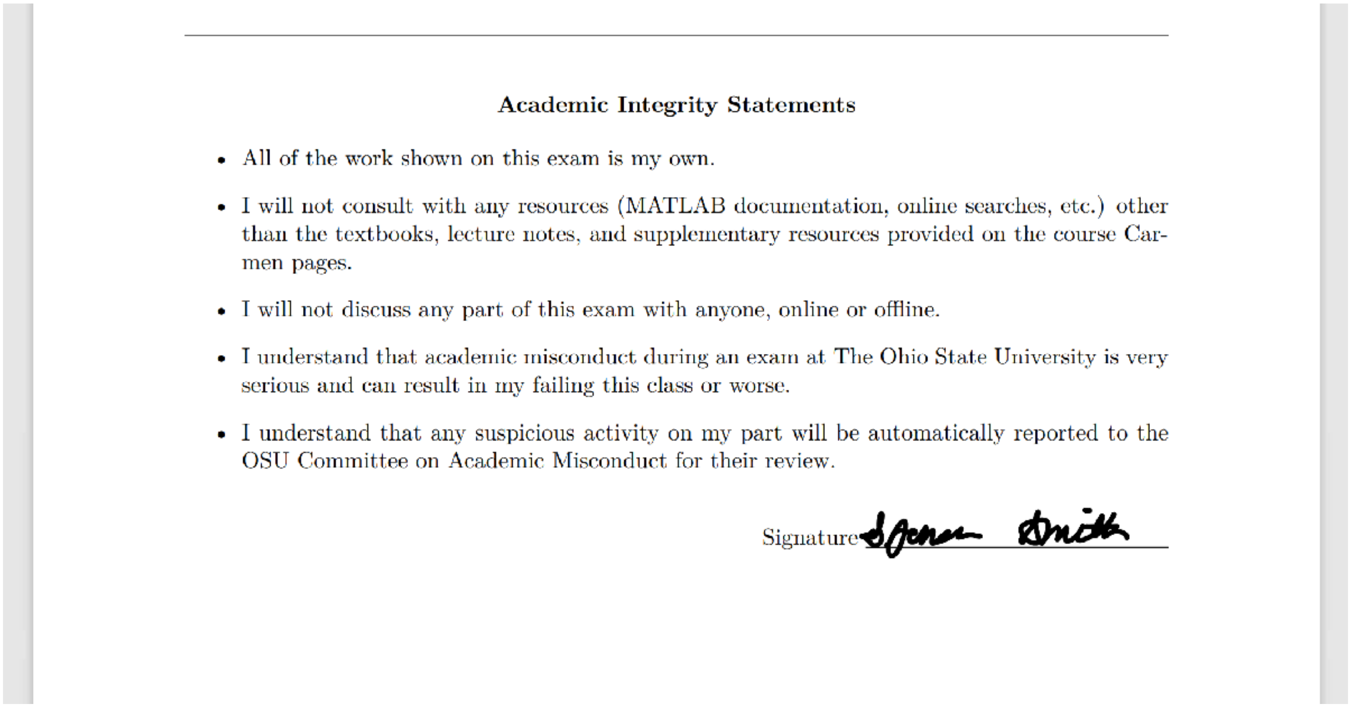
Table of Contents

Instructions

- Rename this file to "exam3_Lastname_Firstname.mlx" (e.g. "exam3_Kim_TaeEun.mlx"). In addition, type in your OSU name.number at the top of this document where it says [NAME.NUMBER]. Failure to do so will result in point deduction.
- Insert your signature (picture file) in the next section. Submission without a signature WILL NOT be accepted.
- Fill in the live script with your answers. Lengthy explanation of your code or justification involving many mathematical symbols may be typed or handwritten. Do whichever is more convenient for you.
- All MATLAB *functions* written for the exam must be included at the end of this file.
- Once completed, re-run the entire live script to ensure that requested outputs and figures are properly generated.
- Export to pdf. Do NOT manually modify the generated pdf file.
- Upload mlx and pdf files. Do not use any external m-files. Your submission must be self-contained.

Signature

Upon reading all the statements on the front page of the exam, please insert your signature (as a picture file) down below. Submission missing signature WILL NOT be accepted.



Academic Integrity Statements

- All of the work shown on this exam is my own.
- I will not consult with any resources (MATLAB documentation, online searches, etc.) other than the textbooks, lecture notes, and supplementary resources provided on the course Carmen pages.
- I will not discuss any part of this exam with anyone, online or offline.
- I understand that academic misconduct during an exam at The Ohio State University is very serious and can result in my failing this class or worse.
- I understand that any suspicious activity on my part will be automatically reported to the OSU Committee on Academic Misconduct for their review.

Signature [illegible]

Problem 3 (Least Squares via SVD)

(d) Complete the function `lsSVD` at the end of the file. Then run the following code block for testing.

```
rng(2021)
m = 3607; n = 21;
A = randn(m, n);
b = randn(m, 1);
x_svd = lsSVD(A, b); x_qr = lsqrfact(A, b);
fprintf(' Results by QR vs SVD\n')
```

```
 Results by QR vs SVD
```

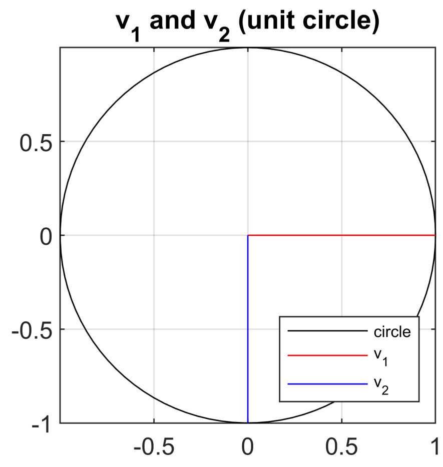
```
fprintf('  absolute error: %10.4e\n', norm(x_svd - x_qr))
```

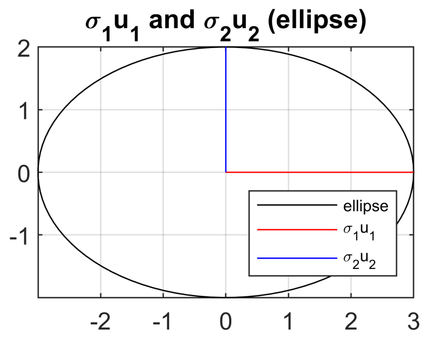
```
  absolute error: 1.3358e-16
```

Problem 4 (Geometric Interpretation of SVD)

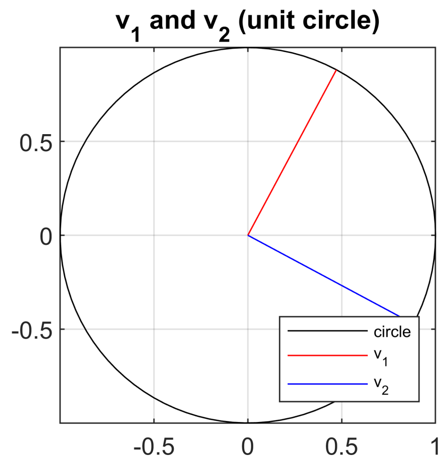
(b)

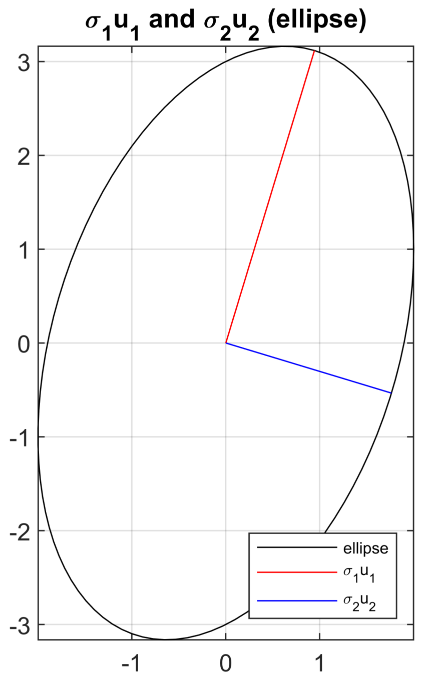
```
A1 = [3 0; 0 -2];
geo_SVD(A1)
```



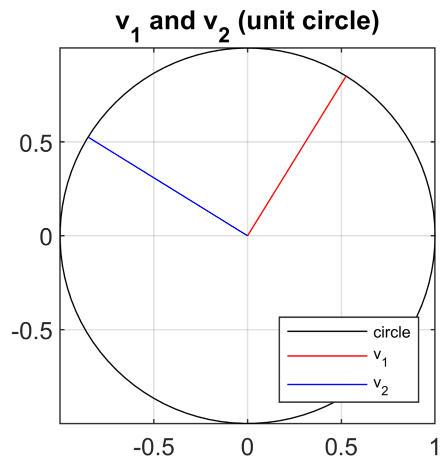


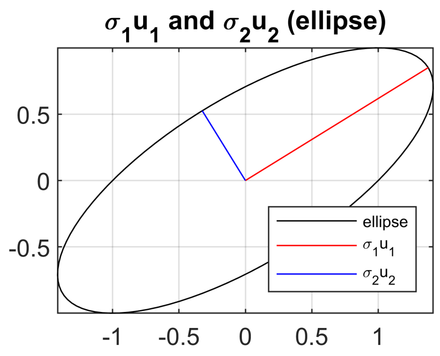
```
A2 = [2 0; 1 3];
geo_SVD(A2)
```





```
A3 = [1 1; 0 1];
geo_SVD(A3)
```

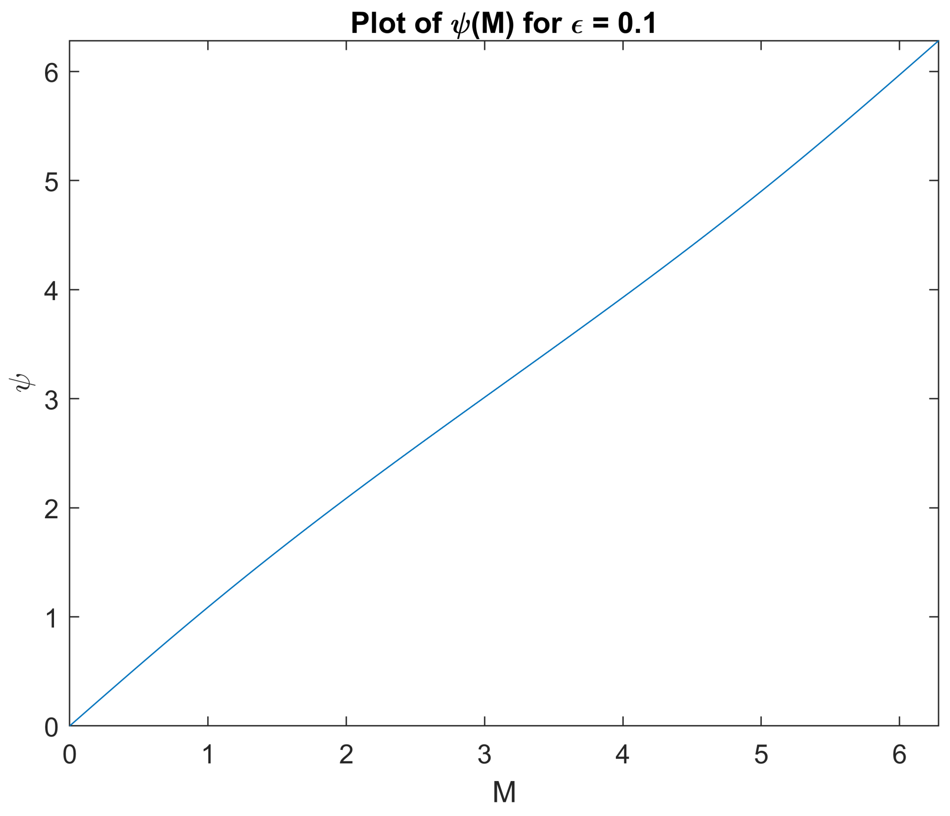




Problem 5 (Asteroids)

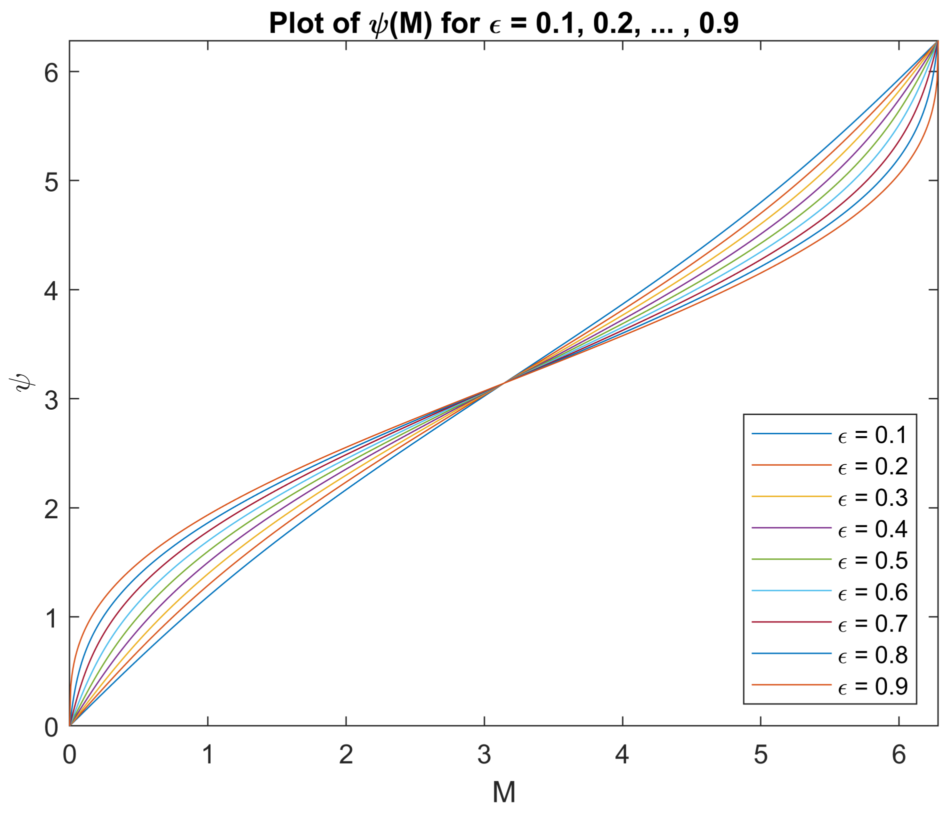
(a)

```
clf
M = linspace(0,2*pi,800);
epsilon = .1;
f = @(psi) psi - (epsilon*sin(psi)); %given function
psi = zeros(800,1); %creates array of zeros for storage
for k=1:800 %800 points so 800 iterations
    g = @(psi) f(psi) - M(k); %given function minus value of M to allow for zeroing
    psi(k) = fzero(g, .1);   %finds what value of psi yields value of M that we are testing
end
plot(M, psi)
axis tight
title('Plot of \psi(M) for \epsilon = 0.1')
xlabel('M')
ylabel('\psi')
```



(b)

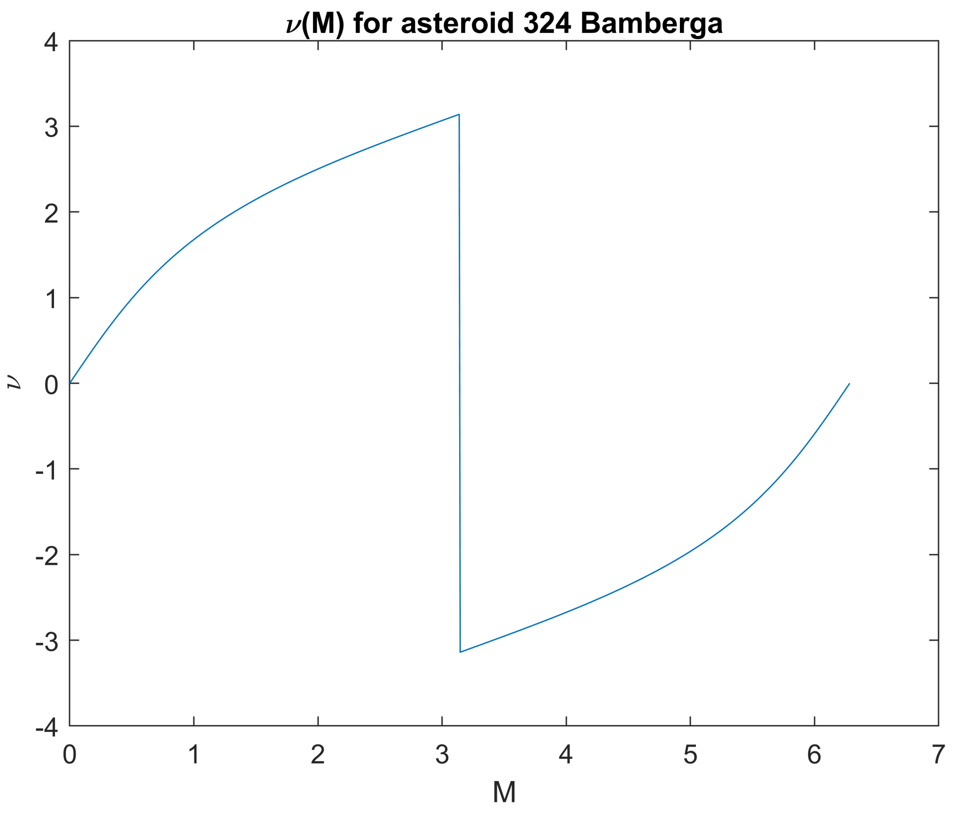
```
clf
M = linspace(0,2*pi,800);
epsilon = .1;
for j=1:9  %need nine iterations b/c we have 0.1,0.2,0.3,...,0.9
    epsilon=epsilon+.1;
    f = @(psi) psi - (epsilon*sin(psi));   %same process as above
    psi = zeros(800,1);
    for k=1:800
        g = @(psi) f(psi) - M(k);
        psi(k) = fzero(g, .1);
    end
    plot(M, psi)
    axis tight
    hold on
end
title('Plot of \psi(M) for \epsilon = 0.1, 0.2, ... , 0.9')
xlabel('M')
ylabel('\psi')
legend({'\epsilon = 0.1', '\epsilon = 0.2', '\epsilon = 0.3', '\epsilon = 0.4', '\epsilon = 0.5
```



(c)

```
clf
eps_bam = .338;
tau_bam = 4.4;
M = linspace(0,2*pi,800);       %same process as above
f = @(psi) psi - (eps_bam*sin(psi));
zero = zeros(800,1);
for k=1:800
    g = @(psi) f(psi) - M(k);
    zero(k) = fzero(g, .1);
end

z = (sqrt((1+eps_bam)/(1-eps_bam))) .* tan(zero/2); %right-side of nu equation
nu = 2*atan(z); %calculation of nu
plot(M, nu)
title('\nu(M) for asteroid 324 Bamberga')
xlabel('M')
ylabel('\nu')
```



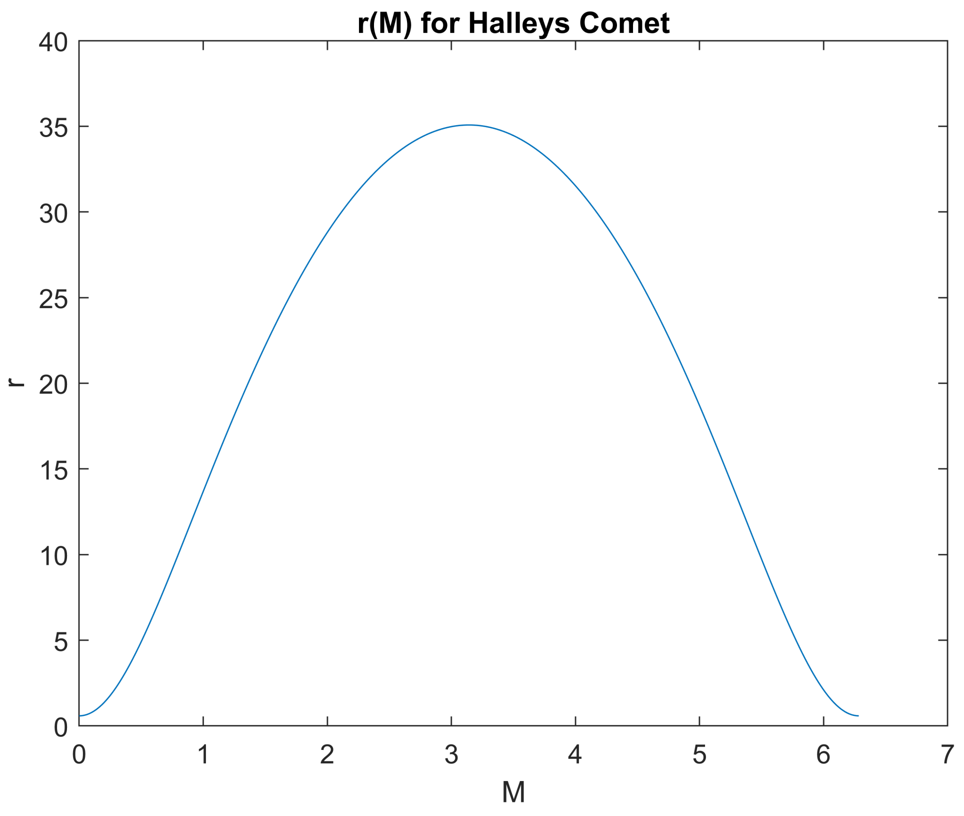
(d)

```
clf
eps_hal=.967;
tau_hal=75.3;
M = linspace(0,2*pi,800);
f = @(psi) psi - (eps_bam*sin(psi));
zero = zeros(800,1);
for k=1:800
    g = @(psi) f(psi) - M(k);
    zero(k) = fzero(g, .1);
end

w = (sqrt((1+eps_hal)/(1-eps_hal))) .* tan(zero/2);
nu = 2*atan(w);   %same calculations as above

a = (39.47524* (tau_hal/(2*pi))^2)^(1/3); %calculation of a

r = (a*(1-(eps_hal^2))) ./ (1 + eps_hal .* cos(nu)); %calculation of r
plot(M, r)
title('r(M) for Halleys Comet')
xlabel('M')
ylabel('r')
```



```
fprintf('The calculated aphelion of Halleys comet is: %f\n', max(r))
```

```
The calculated aphelion of Halleys comet is: 35.074253
```

```
fprintf('The calculated periphelion of Halleys comet is: %f\n', min(r))
```

```
The calculated periphelion of Halleys comet is: 0.588436
```

Functions Used

lsSVD

```
function x = lsSVD(A,b)
% LSSVD x = lsSVD(A,b)
% Solve linear least squares by SVD
% Input:
% A coefficient matrix (m-by-n, m>n)
% b right-hand side (m-by-1)
% Output:
% x minimizer of || b - Ax || (2-norm)
    [U, S, V] = svd(A, 0);       % thin SVD
    z = U' * b;
    x = V' \ (S \ z);    %backward substitution
end
```

lsqrfact

This is a code from lecture; it is needed for Problem 3(d).

```
function x = lsqrfact(A,b)
% LSQRFACT x = lsqrfact(A,b)
% Solve linear least squares by QR factorization
% Input:
%   A    coefficient matrix (m-by-n, m>n)
%   b    right-hand side (m-by-1)
% Output:
%   x    minimizer of || b - Ax || (2-norm)
    [Q,R] = qr(A,0);            % thin QR fact.
    z = Q'*b;
    x = R\z;     % backward substitution
end
```

geo_SVD

```
function geo_SVD(A)
    clf
    [U,S,V] = svd(A); %svd

    %% Plot Unit Circle

    r = 1. ; % radius of circle
    th = linspace(0,2*pi) ;
    x = r*cos(th);
    y = r*sin(th);

    subplot(1,2,1);
    plot(x,y,'k')

    grid on
    axis equal
    axis tight
    hold on

    v1 = V(:, 1); %grabs column 1 of V
    v2 = V(:, 2); %grabs column 2 of V

    plot([0 v1(1)], [0 v1(2)], 'r') %plots v1 vector from origin
    plot([0 v2(1)], [0 v2(2)], 'b') %plots v2 vector from origin

    title('v_1 and v_2 (unit circle)')
    lgd = legend({'circle','v_1','v_2'},'Location','southeast'); %legend in bottom right
    lgd.FontSize = 6; %shrinks size of legend
    hold off


    %% Plot ellipse
    u1 = S(1) * U(:,1); %first column of U times first singular value
    u2 = S(4) * U(:,2); %second column of U times second singular value

    xCenter = 0; %center of ellipse is at origin
    yCenter = 0;
    xRadius = sqrt(u1(1)^2 + u1(2)^2); %calculates radius of semi-axis x
    yRadius = sqrt(u2(1)^2 + u2(2)^2); %calculates radius of semi-axis y
    newx = xRadius * x + xCenter;     %creates x-coords. of ellipse
    newy = yRadius * y + yCenter;     %creates y-coords. of ellipse

    rad = atan(u1(2)/u1(1));        %finds rotation angle for ellipse
    R = [cos(rad) -sin(rad); sin(rad) cos(rad)];  %rotation matrix
    newR = R * [newx; newy]; %rotation matrix times ellipse to get rotated ellipse

    subplot(1,2,2)
    plot(newR(1,:),newR(2,:),'k')

    grid on
    axis equal
    axis tight
    hold on

    plot([0 u1(1)],[0 u1(2)], 'r') %plots u1 vector from origin
    plot([0 u2(1)],[0 u2(2)], 'b') %plots u2 vector from origin

    title('\sigma_1u_1 and \sigma_2u_2 (ellipse)')
    lgd2 = legend({'ellipse','\sigma_1u_1','\sigma_2u_2'}, 'Location', 'southeast');
    lgd2.FontSize = 6;
    hold off
end
```