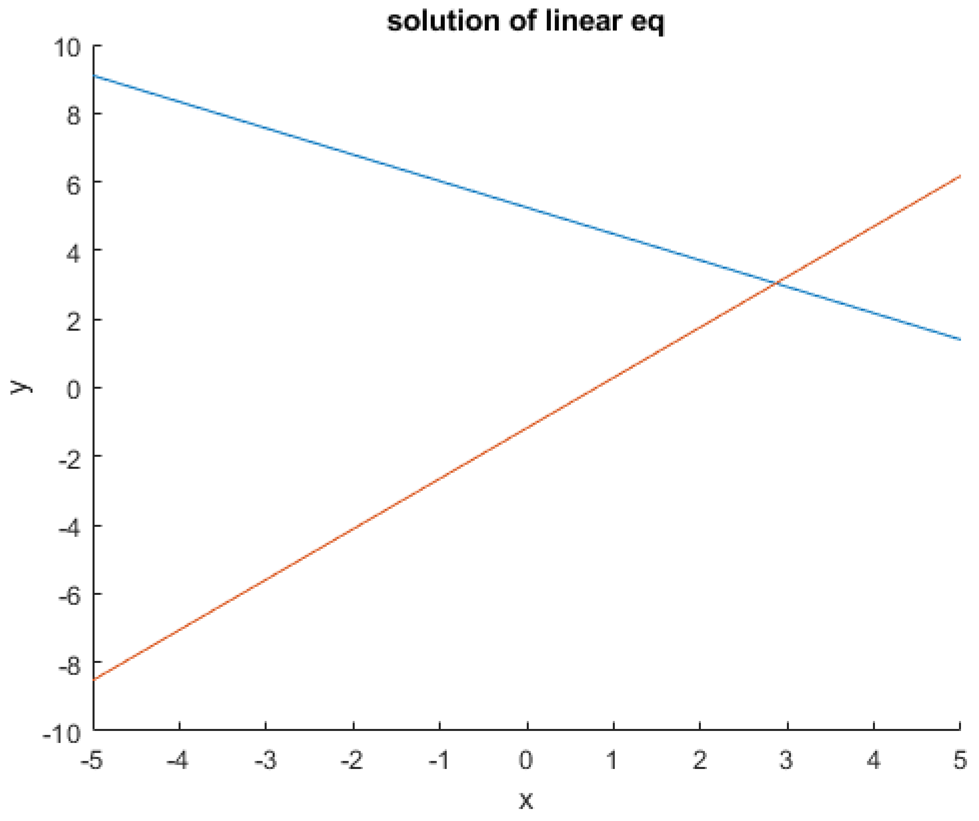
Table of Contents

```
% INSTRUCTIONS:
% 1) save this file into the directory that holds your work
% 2) make sure it runs without error
% 3) publish this file to PDF.
% 4) There should then be a PDF file in the subdirectory 'html'
%    that you can upload to TurnItIn.
%    PLEASE DOUBLE-CHECK it has everything you need!!

% prepare to run...
clear all  % clear out all variables
close all  % close any open figures
```

problem 1

```
type('EquationPair1')

fprintf('\n-----------------------------------\n')
EquationPair1


function [condNum, bCheck] = EquationPair1
%{
Solves linear equation system and outputs the condition
matrix and checks the answer of the solutions as output bCheck

Alyssa Rose  HW7  03-17-18
%}
%sets up coeffecient matrix
A = [0.77,1;2.5,-1.7];
%sets up solution matrix
b = [5.25;2];
sol = A\b;
%prints out solution values of x and y
fprintf('X = %.3f\nY = %.3f\n', sol(1,1),sol(2,1));
%computes condition matrix
condNum = cond(A);
fprintf('Conditional number = %f',condNum);
%checks solution
bCheck = A * sol
```

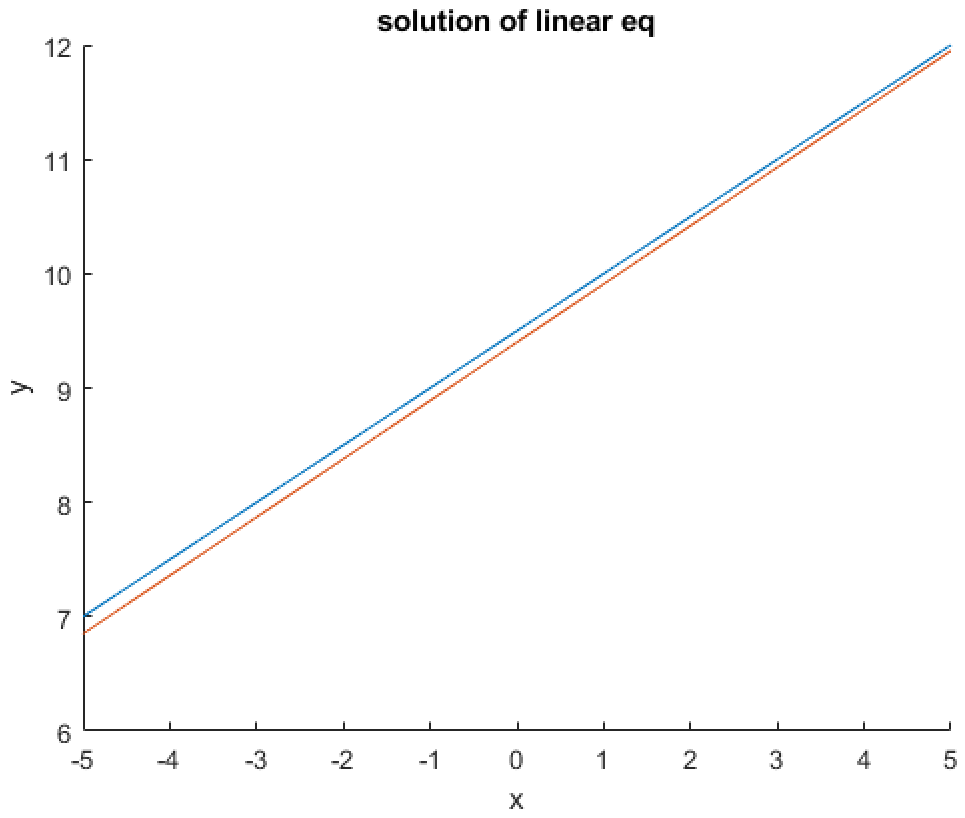
```
%creates symbolic functions of the 2 eqs.
syms x
y1 = -0.77*x + 5.25;
y2 = (2.5*x-2)/1.7;
figure(1)
hold on
fplot(y1);
fplot(y2);
xlabel('x')
ylabel('y')
title('solution of linear eq')
hold off
end
%{
the lines appear to cross close to (3,3), which is approximate
to the solution of the equations, (2.868, 3.041)
%}

----------------------------------
X = 2.868
Y = 3.041
Conditional number = 2.401339
bCheck =

    5.2500
    2.0000


ans =

    2.4013
```



problem 2

```
type('EquationPair2')

fprintf('\n----------------------------------\n')
EquationPair2


function [condNum, bCheck] = EquationPair2
%{
Solves linear equation system and outputs the condition
matrix and checks the answer of the solutions as output bCheck

Alyssa Rose  HW7  03-17-18
%}

%sets up coefficient matrix
A = [0.5,-1;1.02,-2];
%sets up solution matrix
b = [-9.5;-18.8];
sol = A\b;
fprintf('X = %.3f\nY = %.3f\n', sol(1,1),sol(2,1));
%computes condition matrix
condNum = cond(A);
fprintf('Conditional number = %f',condNum);
%checks solutions
```

```
bCheck = A * sol;

%creates symbolic functions
syms x
y1 = 0.5*x + 9.5;
y2 = (1.02*x+18.8)/2;
figure(2)
hold on
fplot(y1);
fplot(y2);
xlabel('x')
ylabel('y')
title('solution of linear eq')
hold off

%{
The condition number for this function is significatnly
higher than that for the first equation (diff between 2.4 and 314).
This could've been predicted based on the graphs also as
equation 1 has a distinct intersection point, whereas equation
2 has lines that run very close to each other (making solving
for their intersection more challenging and more prone to error).
%}
end

----------------------------------
X = 10.000
Y = 14.500
Conditional number = 314.516821
ans =

  314.5168
```



problem 3

```
type('ThreeByFour')


function x = ThreeByFour
%{
finds the solution of linear equations of a non square
matrix

Alyssa Rose  HW7  03-17-18
%}
A = [1,-2,1; 2,-5,3; 1,2,2; 2,0,3];
b = [2,6,4,6]';
x = A\b;
bCheck = A*x;

%{
It is not possible to calculate the determinant of the matrix
since the "det" function in Matlab requires it to be a square
matrix (NxN). Since it isn't, it's not possible to calculate
the determinant.

The values stored in x and the computed values of bCheck match
the b matrix used in the function exactly.
%}
```

```
end
```

problem 4

```
type('NutritionProblem')


function x = NutritionProblem
%{
finds quantity of ingredients to use based on diet restrictions

Alyssa Rose  HW7  03-17-18
%}
matA = ([36,51,13; 52,34,74; 0,7,1]);
b = [33,45,3]';
x = matA\b*100

%{
The units of the solution is in grams
%}
end
```

problem 5

```
type('ComponentsProduced')
fprintf('\n----------------------------------\n')
type('ComponentExamples')


function c = ComponentsProduced(alumKg, plasKg, rubbKg)
%{
takes in kg of aluminum, plastic and rubber and outputs how
many and which componenets can be produced

Alyssa Rose  HW7  03-17-18
%}
%creates coefficient matrix
a = [0.015,0.017,0.019;0.00025,0.00033,0.00042;0.001,0.0012,0.0016];
%creates solution matrix (amount of resources available)
b = [alumKg;plasKg;rubbKg];
%finds answers to solution of linear equations
c = a\b
end

----------------------------------

%{
finds components produced with given resources

Alyssa Rose  HW7  03-17-18
%}

ComponentsProduced(2.12,0.043,0.164);
ComponentsProduced(2.12,0.05,0.164);

%{
the answers for the first call of the function seem reasonable
, and the second does also besides the negative value for the
first solution. Constraints applied to this might have to
make all the solutions positive.
%}
```

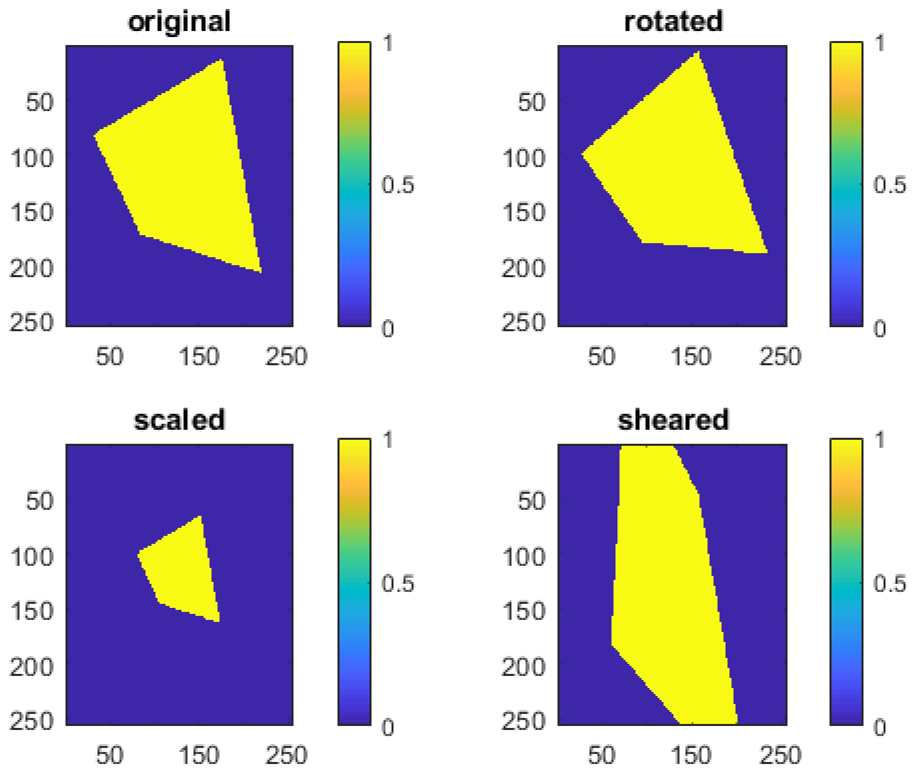
problem 6

```
type('PolygonWarper')
fprintf('\n----------------------------------\n')
type('WarpingScript')
fprintf('\n----------------------------------\n')
WarpingScript


function [polyOut,areaIn,areaOut,detA] = PolygonWarper(polyIn,A)
%{
Inputs:
-polyIn is an N x 2 matrix (N = # of vertices) for input polygon
-A is an 2 x 2 matrix
Outputs:
-polyOut has the N x 2 warped polygon vertices
-areaIn is the area of the input polygon
-areaOut is the area of the warped polygon
-detA is the determinant of A

Alyssa Rose  HW7  03-17-18
%}
%calcs. determinate of A
detA = det(A);
%finds area of original polygon
areaIn = polyarea(polyIn(:,1),polyIn(:,2));
%finds center point of polygon
cenX = sum(polyIn(:,1))/ numel(polyIn(:,1));
cenY = sum(polyIn(:,2))/ numel(polyIn(:,2));
%adjusts polygon to center
polyOut(:,1) = polyIn(:,1) - cenX;
polyOut(:,2) = polyIn(:,2) - cenY;
%warps the polygon by some factor "A" (matrix)
polyOut = polyOut * A;
%shifts polygon back to center
polyOut(:,1) = polyOut(:,1) + cenX;
polyOut(:,2) = polyOut(:,2) + cenY;

%finds area of new polygon
areaOut = polyarea(polyOut(:,1),polyOut(:,2));

imgOut = poly2mask(polyOut(:,1),polyOut(:,2),256,256);
imagesc(imgOut);
colorbar;
```

```
colormap;
%plots original and new polygon using polymask
%commented out to avoid overriding graphs in WarpingScript
%{
subplot(2,1,1)
imgIn = poly2mask(x,y,imageSize,imageSize);
imagesc(imgIn)
colorbar
colormap

subplot(2,1,2)
imgOut = poly2mask(xOut,yOut,imageSize,imageSize);
imagesc(imgOut)
colorbar
colormap
%}
end





----------------------------------

%{
Graphs polygon subjected to different scaling matrices

Alyssa Rose  HW7  03-17-18
%}

load posStarting.mat
%original polygon
subplot(2,2,1)
PolygonWarper(posStarting,eye(2));
title('original')
%rotated polygon
subplot(2,2,2)
A = [cos(10*(pi/180)), -sin(10*(pi/180)); sin(10*(pi/180))
 cos(10*(pi/180))];
[polyOut,areaIn,areaOut,detA] = PolygonWarper(posStarting,A);
title('rotated')
fprintf('rotation: determinant is %.3f, ratio of (new area/old area)
 is %.3f\n', detA , areaOut/areaIn)
%scaled polygon
subplot(2,2,3)
I = [eye(2)*0.5];
[polyOut,areaIn,areaOut,detA] = PolygonWarper(posStarting,I);
title('scaled')
fprintf('scaled: determinant is %.3f, ratio of (new area/old area) is
 %.3f\n', detA , areaOut/areaIn)
%sheared polygon
subplot(2,2,4)
S = [.25,2;.75,.3];
[polyOut,areaIn,areaOut,detA] = PolygonWarper(posStarting,S);
```

```
title('sheared')
fprintf('sheared: determinant is %.3f, ratio of (new area/old area) is
 %.3f\n', detA , areaOut/areaIn)

%{
the absolute value of the ratio of new area to old area and the
 determinant are equivalent
%}
```

```
----------------------------------
rotation: determinant is 1.000, ratio of (new area/old area) is 1.000
scaled: determinant is 0.250, ratio of (new area/old area) is 0.250
sheared: determinant is -1.425, ratio of (new area/old area) is 1.425
```



Problem 7 needs to be added by PDF merge

Published with MATLAB® R2017b