   b) Do problem 9 from section 6.1 (pg. 284).


Problem 3: Consider the matrix

\[
M = \begin{bmatrix}
2 & 2 & 1 & 1 \\
-14 & -6 & -9 & -7 \\
-2 & -1 & -2 & -1 \\
8 & 1 & 7 & 4
\end{bmatrix}
\]

a) One eigenvector is \(x_1 = (1, 1, 0, -3)\). What is the corresponding eigenvalue?
   b) Note that \(\det(M) = 0\). Use this information to find another eigenvalue \(\lambda_2\) - how do you know this must be an eigenvalue?
   c) A third eigenvalue is \(\lambda_3 = -1\). Write down (but don’t solve) a linear system that can be solved to find \(x_3\).
   d) What is the fourth eigenvalue? (Hint: use the trace.)

Problem 4: a) Do problem 8 in section 6.2 (pg. 299)
   b) Do problem 18 in section 6.2 (pg. 300)

Problem 5: Here’s an example of an invertible 3 by 3 matrix with only 2 different eigenvalues:

\[
A = \begin{bmatrix}
4 & 1 & -1 \\
2 & 5 & -2 \\
1 & 1 & 2
\end{bmatrix}
\]

a) Find the eigenvalues of \(A\).
   b) Find 3 linearly independent eigenvectors of \(A\).
   c) Is \(A\) diagonalizable? If so, write down a diagonalization \(A = S\Lambda S^{-1}\).

Problem 6: Do problems 15 and 16 in section 6.2 (pg. 300).
Problem 7: Do problem 22 in section 6.2 (pg. 301).

Problem 8: Do problem 7 in section 8.3 (pg. 429).

Problem 9: Do problem 8 in section 8.3 (pg. 429).

Problem 10: A Matlab question: The page rank algorithm in Google is essentially solving an eigenvalue problem for a matrix \( M \) with size in the billions. The method is discussed on pages 358-359 of the textbook; you can find more information in an article by Cleve Moler (MATLAB founder):

www.mathworks.com/company/newsletters/news_notes/clevescorner/oct02_cleve.html

The idea is to start crawling randomly from a website and count the frequency of hitting each site. We create an adjacency matrix that represents the links between websites. By rescaling the columns, we obtain a Markov matrix \( M \) - it tells us the probability of getting to a website by following a random link. If we act by \( M \) repeatedly, vectors will tend to the steady state vector. We’ll call this the evector. The evector represents the total frequency of links to a site, and so sites with larger entries should have higher page ranks. Google finds the evector by crawling randomly through sites.

Model this with a 6 by 6 Markov matrix \( M \) and print the output:

\[
\begin{align*}
W &= \text{ceil}(\text{rand}(6) - .55*\text{ones}(6)) \quad \% \text{create a 1-0 web link matrix } W \\
M &= W*\text{diag}(1./\text{sum}(W)) \quad \% \text{Markov with column sums } = 1 \quad \text{Check sum}(M) \\
[S,L] &= \text{eig}(M) \quad \% S = \text{eigenvector matrix of } M \text{ and } L = \text{eigenvalues} \\
x &= S(:,1); v = x/\text{sum}(x) \quad \% \text{first column is usually evector } v > 0 \text{ for evalue } = 1
\end{align*}
\]

Start from the first website:

\[
u = [1,0,0,0,0,0]’
\]

Now \( Mu \) is the first column of \( M \). Using the column \( Mu \), figure out the probabilities of reaching site 1 to 6.

Define a vector \( f \) that is the fraction of times you hit each of the websites as you continue to crawl. I think \( f \) should approach the evector \( v \) if you act by \( M \) enough times. Does it?