Joseph Free
MATH 3261

1 Problem

In this project we seek to determine the dominant (or largest) eigenvalue and corresponding eigenvector of the matrix

$$A = \begin{bmatrix} 4 & 1 & 1 & 1 \\ 1 & 3 & -1 & 1 \\ 1 & -1 & 2 & 0 \\ 1 & 1 & 0 & 2 \end{bmatrix}.$$

In general, if A is an $n \times n$ matrix and x is an $n \times 1$ vector, we seek all scalars λ and vectors x such that the equation

$$Ax = \lambda x \tag{1}$$

holds. In such a situation, we say the scalar λ is an eigenvalue of A, and x is the eigenvector corresponding to λ, respectively.

In order to determine λ and x, we make the requirement that $x \neq 0$, and observe that solving $Ax = \lambda x$ is equivalent to solving

$$(A - I\lambda)x = 0. \tag{2}$$

Then, from the assumption that $x \neq 0$, we conclude that the matrix $A - I\lambda = 0$, and hence must be singular. Therefore, it follows that $det(A - I\lambda) = 0$.

Since A is $n \times n$, from the computation of the determinant we obtain an nth degree polynomial in λ, which we call the characteristic polynomial of A. The roots of this polynomial are precisely the eigenvalues of A. Once we've obtain a given eigenvalue, we need only return to equation (2) in order to determine the corresponding eigenvector.

It is easy to see that one method of obtaining the dominant eigenvalue of our matrix A is finding the roots of its characteristic polynomial and ordering them by size. However, it is just an easy to see that as the size of A increases, so too does the difficulty of finding the roots of the characteristic polynomial. Since we are dealing with a 4×4 matrix, we would expect to be finding the roots of a fourth degree polynomial, which might be tedious. Thus, we will utilize an iterative method known as the Power Method to approximate the dominant eigenvalue.

In order to proceed with the Power Method, we make two basic assumptions about the eigenvalues and eigenvectors. Namely, that the eigenvalues of A may be ordered in such a way that $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$ and the corresponding eigenvectors u_i, $1 \leq i \leq n$, form a linearly independent set $\{u_1, u_2, \ldots, u_n\}$.

Since the set of eigenvectors of A is linearly independent, it is possible to express any vector $x \in \mathbb{R}^n$ as a linear combination of the vectors u_i. Thus,

$$x = \sum_{j=1}^{n} b_j u_j.$$

Then by successive multiplication by the matrix A, we can generate the sequence

$$\begin{array}{rcccc} y_0 = & Ax & = & \sum_{j=1}^n b_j A u_j & = \sum_{j=1}^n b_j \lambda_j u_j \\ y_1 = & A^2x & = & \sum_{j=1}^n b_j A^2 u_j & = \sum_{j=1}^n b_j \lambda_j^2 u_j \\ y_3 = & A^3x & = & \sum_{j=1}^n b_j A^3 u_j & = \sum_{j=1}^n b_j \lambda_j^3 u_j \\ & \vdots & & & \\ & \vdots & & & \\ y_k = & A^kx & = & \sum_{j=1}^n b_j A^k u_j & = \sum_{j=1}^n b_j \lambda_j^k u_j \end{array} .$$

By factoring out λ_1^k out of the right hand side of the last equation, we obtain

$$y_k = \lambda_1^k \sum_{j=1}^{n} b_j \left[\frac{\lambda_j}{\lambda_1}\right]^k u_j. \tag{3}$$

Expanding (3), we see that

$$y_k = \lambda_1^k \sum_{j=1}^{n} b_j \left[\frac{\lambda_j}{\lambda_1}\right]^k u_j. = \lambda_1^k \left(b_1 u_1 + b_2 \left[\frac{\lambda_2}{\lambda_1}\right]^k u_2 + \ldots + b_n \left[\frac{\lambda_n}{\lambda_1}\right]^k u_n \right)$$

It stands to reason now that since λ_1 is the largest of all the eigenvalues of A, the quantity $\left[\frac{\lambda_j}{\lambda_1}\right]^k$ must be less than one. Thus, if we take the limit as $k \longrightarrow \infty$, all the terms in (3) but the first must go to zero, and so we conclude that

$$y_k \longrightarrow \lambda_1^k b_1 u_1 \qquad as \qquad k \longrightarrow \infty$$

This is the general theory behind the power method. With slight modifications on this general theme, we can obtain a number of ways to approximate λ_1. In the source code section below, we implement one such way of doing so using an algorithm from Burden and Faires Numerical Analysis 8th Edition.

2 Source Code

File Name: pMethod_hw6.m
Assignment: Project 6
Student: Joseph Free
Course: MATH3261

Purpose: This function implements the Power Method as given in Burden and Faires 8th Edition. Given an n $\times$ n matrix A, it calculates the largest eigenvalue of A and find the corresponding eigenvector.

Required Input:

```
A    --- The n by n matrix in question.
x    --- An initial guess vector.
TOL  --- Required tolerance.
N    --- Maximum number of iterations.

function [ u, x ] = pMethod_hw6( A, x, TOL, N )

k = 1;
p = find( max( abs(x) ) );

x = x/max(x);

while( k < N )

y = A*x;
u = y(p);
p = find( max( abs(y) ) );

if ( u == 0 )
fprintf('Eigenvector:\n ')
disp(x)

fprintf('A has eigenvalue 0, pick a new initial vector.');
break
end

ERR = max ( abs( x - y/y(p) ) );
x = y/y(p);
```

```
if ( ERR < TOL )
break
end

k = k+1;

end

if ( k >= N )
fprintf('Maximum number of iterations exceeded.')
end

end
```

File Name: main_hw6.m
Assignment: Project 6
Student: Joseph Free
Course: MATH3261

Purpose: This script drives the pMethod_hw6 function. All results for this project come from this file.

```
A = [ 4 1 1 1; 1 3 -1 1; 1 -1 2 0; 1 1 0 2 ]

x0 = [1; -2; 0; 3]

[u, x] = pMethod_hw6(A, x0, .0001, 20)

[u, x] = pMethod_hw6(A, x0, .0001, 25)

[u,x] = pMethod_hw6(A, [1,0,1,0]', .0001, 30)
```

3 Output/Results

```
A =

     4     1     1     1
     1     3    -1     1
     1    -1     2     0
     1     1     0     2


x0 =

     1
    -2
     0
     3

( pMethod(A, x0, .0001, 20 )
Maximum number of iterations exceeded.
u =

    5.2348

x =

    1.0000
    0.6168
    0.1189
    0.4995

( pMethod(A, x0, .0001, 25 ) )
u =

    5.2359
x =

    1.0000
    0.6178
    0.1182
    0.4999
```

```
( pMethod(A, x0, .0001, 30 ) )
u =

    5.2359


x =

    1.0000
    0.6178
    0.1182
    0.4999
```

4 Conclusion

From the results section above, we see that given A as defined at the beginning of this paper and initial guess vector $x_0 = (1, -2, 0, 3)^t$, the power method finds $\lambda_1 \approx 5.2359$ with corresponding eigenvector $x \approx (1.0000, .6178, .1182, .4999)^t$. This value is within .0001 of the actual dominant eigenvalue of A. Using a completely different initial guess ($x_0 = (1, 0, 1, 0)^t$) at the same level of tolerance, the method converges to exactly the same eigenvalue/eigenvector.
Furthermore, as evidenced by the first call to pMethod_hw6, we note that the method takes no less than 20 iterations to reach the desired level of accuracy in this particular case.