MATLAB Exercise • Level 2

CHARLES J. AMMON
DEPARTMENT OF GEOSCIENCES
PENN STATE UNIVERSITY

Eigenvalues & Eigenvectors of A Symmetric Tensor

EIGENVALUES AND EIGENVECTORS

Eigen-analysis is a broad, important branch of continuous and discrete mathematics that frequently is encountered in applied mathematics. For this exercise we are going to focus on the computation of the eigenvalues and eigenvectors of a matrix. MATLAB is designed for this type of analysis and is a natural tool for exploring the concepts of eigenvectors and eigenvalues. The significance of the eigenvalues depends on the particular problem you are working on.

To make our example more concrete, we'll examine problem where we want to compute the orientation and magnitudes of the principal stresses corresponding to a cartesian stress tensor

$$\sigma = \begin{bmatrix} \sigma_{nn} & \sigma_{ne} & \sigma_{nd} \\ \sigma_{en} & \sigma_{ee} & \sigma_{ed} \\ \sigma_{dn} & \sigma_{de} & \sigma_{dd} \end{bmatrix} \tag{1}$$

where n, e, d represent north, east, and down. From physical considerations, we also have $\sigma_{ij} = \sigma_{ji}$, which means that the tensor is symmetric. Physically, in this case the eigenvalues represent the magnitudes and the eigenvectors represent the normals to planes that would experience only norm stresses in a region where the stress is specified by σ_{ij}.

The Eigenvalue Equations

The equations for calculating the eigenvalues of a matrix, S, are solutions of the equation

$$S\vec{x} = \lambda\vec{x}. \tag{2}$$

Equation (2) provides one way to check our answers, when we compute an eigenvalue and an eigenvector, we can substitute the pair into (2) and insure that the equality holds. Although I believe that it is critical for a student to learn how to compute the eigenvalues and eigenvectors of a small matrix without a tool such as MATLAB, the point of this note is to show how to do the job with MATLAB - which makes it very easy because there exists a special command to solve our problem.

EIGENVALUES AND EIGENVECTORS FROM MATLAB

To illustrate the process, we'll need a sample stress tensor. Let

$$\sigma = \begin{bmatrix} 100 & -50 & 0 \\ -50 & 150 & -35 \\ 0 & -35 & 30 \end{bmatrix} \cdot MPa \tag{3}$$

where MPa indicates units of mega-pascals.

Here's my script to do the decomposition. The eigenvalues are computed in one line, the rest of the script is simply sorting the values and making a neater output.

```
function [V, D] = stensor_decomp(S)
%
%  function to decompose a symmetric tensor, S,
%
%  INPUT:
%      the tensor, S in the form:
%
%           | Snn Sne Snd |
%           | Sen See Sed |
%           | Sdn Sde Sdd |
%
%       where n = north, e = east, and d = down
%
%  into principal values and principal axes
%
%  OUTPUT:
%           matrices V (columns are eigenvectors)
%                    D (diagonal with principal values)
%
%  Example:
%
%     >> S = [100,-50,0;-50,150,-35;0,-35,30];
%     >> [V D] = stensor_decomp(S);
%

% compute the eigenvalues and eigenvectors of S
%
[V0, D0] = eig(S); % That was quick but we're done
%                    except for pretty output
%
% that's almost all the work, but let's sort our results
%    so that the largest value is S1 > S2 > S3
%
[Y, I] = sort(diag(D0));
%
D = diag( [D0(I(3),I(3)), D0(I(2),I(2)), D0(I(1),I(1))] );
V = [ V0(:,I(3)), V0(:,I(2)), V0(:,I(1))];
%
% this is really just some code to print the results
%    in an easier to read format
```

```
%
% Print the eigenvalues
%
format short
mystring = sprintf('\nThe eigenvalues are:\n');
disp(mystring);
mystring = sprintf('     %6.2f %6.2f %6.2f\n',diag(D));
disp(mystring);
%
%  Print the eigenvectors
%
mystring = sprintf('The eigenvectors are:\n');
disp(mystring);
mystring = sprintf('        V1     V2    V3\n');
disp(mystring);
mystring = sprintf('  N: %6.3f %6.3f %6.3f\n',V(1,:));
disp(mystring);
mystring = sprintf('  E: %6.3f %6.3f %6.3f\n',V(2,:));
disp(mystring);
mystring = sprintf('  D: %6.3f %6.3f %6.3f\n',V(3,:));
disp(mystring);
%
```

To use the script, as you can see in the comments, you execute

```
>> S = [100,-50,0;-50,150,-35;0,-35,30];
>> [V, D ] = stensor_decomp(S);
>>
>> The eigenvalues are:
>>
>>      186.67  75.39  17.94
>>
>> The eigenvectors are:
>>
>>         V1     V2    V3
>>
>>   N:  0.491  0.849 -0.195
>>
>>   E: -0.850  0.418 -0.319
>>
>>   D:  0.190 -0.322 -0.927
```

which achieves our goal. We can see that the principal axis of this stress tensor points in a west-northwest direction and slightly downward. The azimuth could be computed using MATLAB's *atan2* function (and accounting for the difference in reference direction, north is along the vertical axis).

```
>> v1 = V(:,1);
>> »az1 = (180/pi)*(pi/2 - atan2(v1(1),v1(2)))
>> az1 =
>>   -60.0181
```

EXERCISES

Exercise 1: You can always check the decomposition of a symmetric matrix by reconstructing the original matrix from the eigenvalues and eigenvectors using

$$S = VDV^{\dagger} \tag{4}$$

where V is the matrix whose columns are the eigenvectors of S, D is a diagonal matrix composed of the eigenvalues, and $V^{\dagger}$ is the transpose of V. Use the script to decompose a symmetric tensor and then reconstruct the original tensor using equation (4).

Exercise 2: Test each of the eigenvalues and eigenvectors in the defining equations of the eigenvalue problem (equation (2)).