Class Project :: Part Four

Create a new directory, '`class project/part4`', for your matrix library code and copy your class project fortran files from '`part3`' into it.

- Write a Function called '`transmat`' to return the transpose of a matrix. The transpose of a matrix $\mathbf{A}$ is written $\mathbf{A}^{\mathrm{T}}$ and is simply $\mathbf{A}$ with the rows and columns interchanged. So row one in $\mathbf{A}^{\mathrm{T}}$ is column one in $\mathbf{A}$ etc.

- Write a Function called '`normtwo`' to return the Euclidean norm, ($\| \mathbf{v} \|_2$) of a vector $\mathbf{v}$. The Euclidean norm is often referred to as the length of a vector and is the square root of the sum of the squares of all the elements in the vector.

The Power Method

Let $\mathbf{A}$ be an $m \times m$ real matrix with m real eigenvalues. Moreover, assume that $\mathbf{A}$ has precisely one eigenvalue (λ_1) that is 'dominant' (largest in absolute magnitude) with a corresponding eigenvector ($\mathbf{x}$). Then $\mathbf{x}$ and λ_1 can be calculated using the 'power method' for approximating eigenvalues. The 'power method' is an iterative method that, for each iteration, calculates a new estimate of the eigenvector $\mathbf{x}^{(n+1)}$ from the previous estimate $\mathbf{x}^{(n)}$.

[Q] How does the 'power method' work?

[A] It works by taking an initial 'guestimate' at the eigenvector and then repeatedly premultiplying this guess with the matrix $\mathbf{A}$. The m eigenvectors of our $m \times m$ real matrix $\mathbf{A}$ form a set of linearly independent basis vectors in $\mathbb{R}^m$ $\{\mathbf{v}_1, \mathbf{v}_2, \dots , \mathbf{v}_m\}$. The key to understanding this method is by making use of the fact that the arbitrary vector $\mathbf{x}$ (our initial guess) can be written as a linear combination of these m eigenvectors. Therefore,

$$\begin{aligned}
\mathbf{x} &= c_1\mathbf{v}_1 + c_2\mathbf{v}_2 +, \dots\dots, +c_m\mathbf{v}_m && (1)\\
\mathbf{A}\mathbf{x} &= c_1\lambda_1\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 +, \dots\dots, +c_m\lambda_m\mathbf{v}_m && (2)\\
\mathbf{A}(\mathbf{A}\mathbf{x}) = \mathbf{A}^2\mathbf{x} &= c_1\lambda_1^2\mathbf{v}_1 + c_2\lambda_2^2\mathbf{v}_2 +, \dots\dots, +c_m\lambda_m^2\mathbf{v}_m && (3)\\
\mathbf{A}^n\mathbf{x} &= c_1\lambda_1^n\mathbf{v}_1 + c_2\lambda_2^n\mathbf{v}_2 +, \dots\dots, +c_m\lambda_m^n\mathbf{v}_m && (4)
\end{aligned}$$

The equations $1 \rightarrow 4$ show how repeatedly premultiplying $\mathbf{x}$ with $\mathbf{A}$ builds up the m terms involving the m eigenvectors. Now since the first term on the RHS of (4) contains λ_1, the dominant (largest in absolute value) eigenvalue, as n gets larger this first term will start to dominate the other terms in absolute value. So the RHS will eventually converge to our dominant eigenvector as the first term containing $\mathbf{v}_1$ is providing the largest contribution to the summation. There is a problem, however, that as n gets large λ^n gets very large in fact too large for the computer to accurately represent its value. So, to solve this problem, after each time we premultiply by $\mathbf{A}$ we normalise the result to keep our new estimate of the dominant eigenvector of order one. Remember that dividing an eigenvector by a scalar still leaves the same eigenvector but simply rescaled.

The main iterative algorithm of power method can be written as;

- Take a normalised estimate of the eigenvector $\mathbf{x}^{(n)}$ and calculate $\mathbf{y}^{(n)} = \mathbf{A} \cdot \mathbf{x}^{(n)}$.

- Calculate the new estimate of the dominant eigenvalue $\lambda_1^{(n)} = \mathbf{y}_k^{(n)} / \mathbf{x}_k^{(n)}$, where $k \in \{1, 2, .., m\}$

- Calculate the new normalised $(n+1)^{th}$ estimate of the dominant eigenvector from the $(n)^{th}$ estimate of $\mathbf{y}$

$$\mathbf{x}^{(n+1)} = \frac{\mathbf{y}^{(n)}}{\| \mathbf{y}^{(n)} \|_\infty}$$

- Repeat the above three steps until some termination criterion has been satisfied. Note that $\mathbf{x}^{(n)}$ means the n^{th} iterative estimate **not** $\mathbf{x}$ raised to the power n.

(a) In your matrix library' `MODULE`' add the Fortran 90 procedures, described below.

(i) First write a '`FUNCTION infnorm(vec)` ' that calculates and returns the 'infinity' norm of a vector $\mathbf{v} \in \mathbb{R}^m$. The infinity norm, $\| \mathbf{v} \|_\infty$, is simply the modulus of the element in $\mathbf{v}$ with the largest absolute value.

(ii) Write a '`FUNCTION cont(y,x,tol,max_iters)`' that returns '`.TRUE.`' if a given tolerance has been met or if the maximum number of iterations has been exceeded otherwise it returns '`.FALSE.`' Use for the tolerance condition,

$$\| \mathbf{x}^{(n+1)} - \mathbf{x}^{(n)} \|_\infty < \text{tol}$$

use tol $= 0.001$.

HINT : Declaring a variable inside a procedure with the attribute '`SAVE`', means that the value of that variable will be retained between calls to that procedure.

```
INTEGER, SAVE ::  count=1
```

declares '`count`' to be initialised with the value '`1`' only on the first call to the procedure it is declared in, from then on it retains its value between calls. So '`count=count+1`' could be used to count the number of calls to the subroutine.

(iii) Write a '`FUNCTION power(mat,x,tol,eigv,conv,max_iters)`' where '`mat`' is the matrix for which the dominant eigenvalue is to be found, '`x`' is to input the initial eigenvector estimate and return the final estimate, '`tol`' is the measure of convergence, '`eigv`' is to hold the returned eigenvalue and '`conv`' is of type '`LOGICAL`' and returns '`.TRUE.`' only if the method converged. '`max_iters`' is the maximum number of iterations the method is to take before giving up and returning to the calling program unit. The '`power`' function should return the number of iterations taken.

(iv) You will need to write function '`mulmatvec(mat,vec) that pre-multiplies a vector by a matrix`'

(b) Write a main program unit that uses your matrix module to calculate the dominant eigenvalue and corresponding eigenvector of the matrix;

$$\begin{pmatrix} 1 & 5 & 3 \\ 6 & 3 & 5 \\ 2 & 8 & 5 \end{pmatrix}$$

In your main program set the maximum number of iterations your code should take as '`500`' and the tolerance '`tol`' to be 0.001.

Your program should report if there was a successful convergence and if so report the dominant eigenvalue and corresponding eigenvector of the matrix along with the number of iterations taken by the method.

You should have a dominant eigenvalue of 13.063. with a corresponding eigenvector of $(0.6039, 0.8569, 1.0000)$.