Numerical Methods 35006
Computer Lab 8: Linear Algebra

For this Lab we will be getting using the `scipy` linear algebra package. This can be imported using the command

```
from scipy import linalg as la
```

Functions and attributes within the package can then be called using `la`, e.g. the inverse of a matrix A can be computed with the command `la.inv(A)`.

---

1. Use python to create the following row and column vectors as `numpy` arrays:

$$\mathrm{a} = [1, 0, 2, 3] \quad , \quad \mathrm{b} = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}$$

Use the `.shape` attribute to confirm that you have defined both vectors correctly.

Compute :

a) The dot product between `a` and `b` using the `@` operator

b) The dot product between `a` and `b` using `a.dot(b)`

c) The transpose of `a`

d) The quantity $|\mathrm{b}|^2 = \mathrm{b}^\mathrm{T} \cdot \mathrm{b}$

e) The L_2 norm of `a`, using `la.norm(a,2)`.

2. An $n \times m$ matrix of random numbers can be computed using the command

```
A = np.random.rand(n,m)
```

Create a random 4×4 matrix A. Compute:

a) The transpose of A

b) The matrix product A b, where b is as in the previous question

c) The inverse of A using `la.inv`, then confirm that this is the inverse by computing the matrix product AA^{-1} and calculating the L_2-norm difference between this and the 4×4 identity matrix given by `np.eye(4)`.

3. Compute a random 20×20 matrix A and a random 20×1 column vector b. By finding the inverse of A, solve the equation

$$\mathrm{Ax} = \mathrm{b}$$

Perform the matrix multiplication A x and compare it with b to verify your solution.

4. Use python to solve the following system of linear equations:

$$\begin{array}{cccc} -F_1 & -3F_2 & +2F_3 & = 4 \\ 3F_1 & -6F_2 & +2F_3 & = 3 \\ & 2F_2 & -F_3 & = 2 \end{array} \qquad (1)$$

5. Compute a random 4×4 matrix A. The LU decomposition of A can be computed using the command

```
P,L,U = la.lu(A)
```

Compute `L` and `U` and verify that they have the right form.

6. The QR decomposition of A can be computed using `la.qr`. Compute the QR decomposition of your matrix A in the previous question. Verify that:

a) R is an upper triangular matrix

b) Q is an *orthogonal matrix*, that is, $\mathrm{Q}^\mathrm{T} = \mathrm{Q}^{-1}$.

7. Generate a random 5×5 matrix, and write code that uses the Power method to compute the largest eigenvalue and eigenvector of this matrix. Test your eigenvalue and eigenvector to make sure that you have correctly solved the eigenvalue problem.

8. Create a function

   ```
   lam,vec = peig(A,tol)
   ```

   that implements the Power method for an arbitrary square matrix A, returning the smallest eigenvalue `lam` with its corresponding eigenvector `vec` when the eigenvalue has converged to within a tolerance `tol`. Test this code, then save it to a new module `myeigs.py`.

9. Create a function that implements the Inverse Power method with the following function call:

   ```
   lam,vec = invpeig(A)
   ```

   Test this code, then save it to your `myeigs` module.

10. Create a function that uses QR iteration to compute all the eigenvalues of a matrix to within a given tolerance. The function should be called using the function call

    ```
    lam = qreigs(A,tol)
    ```

    Test this code and save it to your `myeigs` module.