```
            complex(8), intent(in) :: x(:)
            real(8) :: norm
            norm = sqrt(sum(abs(x)**2))
        end function norm
        ! calculate the Rayleigh quotient
        function rayleigh_quotient(A,x)
            complex(8), intent(in) :: A(:,:), x(:)
            complex(8) :: rayleigh_quotient
            rayleigh_quotient = dot_product(x,matmul(A,x))
            rayleigh_quotient = rayleigh_quotient/dot_product(x,x)
        end function rayleigh_quotient
end program poweriteration
```

Example 7.2 Power iteration (Python)

Use the normalised power iteration to find the dominant eigenvalue and associated eigenvector of the Hermitian matrix

$$A = \begin{bmatrix} 4 & -i & 2 \\ i & 2 & 2+7i \\ 2 & 2-7i & -2 \end{bmatrix}$$

Solution: Before converting it into a Python program, how do we know when to stop the iteration? Since we know that the Rayleigh quotient should converge to the dominant eigenvalue, we can use a while loop, and exit the loop once successive Rayleigh quotients differ by a small enough quantity:

```
#!/usr/bin/env python3
import numpy as np

# function to calculate the Rayleigh quotient
def rayleigh_quotient(A,x):
    return np.dot(x, np.dot(A, x))/np.dot(x,x)

# function to normalise a vector
def normalise(x,eps=1e-10):
    N = np.sqrt(np.sum(abs(x)**2))
    if N < eps: # in case it is the zero vector!
        return x
    else:
        return x/N

A = np.array([[4, -1j, 2],
              [1j, 2, 2+7j],
              [2, 2-7j, -2]])

# choose the starting vector
```

```
x = normalise(np.array([1, 1, 1]))
RQnew = rayleigh_quotient(A,x)
RQold = 0

# perform the power iteration
while np.abs(RQnew-RQold) > 1e-6:
    RQold = RQnew
    x = normalise(np.dot(A, x))
    RQnew = rayleigh_quotient(A, x)

print("Dominant eigenvector:",x)
print("Dominant eigenvalue: {:.5f}".format(RQnew))
```

Problem

The example above returns a dominant eigenvalue of $\lambda_1 = 8.45188$, and dominant eigenvector

$$\mathbf{v}_1 = (0.3398 - 0.2345, 0.4913 + 0.5107i, 0.5011 - 0.2762i).$$

Have a go running the code snippets to verify this result.

Next, try modifying the above example to count how many loop iterations occur before convergence. Is it larger or smaller than you had expected?

An alternative to using the Rayleigh quotient to determine whether convergence is achieved is the convergence criterion

$$|x_{n+1} - x_n| \leq \epsilon.$$

This is especially useful if we do not need to compute the eigenvalue, and can save some computational time. Modify the code above to use this convergence criterion, and compare the overall wall time to the original version.

Google PageRank

Everyday you are likely interacting with a company that uses the power iteration — Google! To determine which websites are more useful when presenting you with search results, Google uses a **network centrality** algorithm called PageRank. By representing internet sites as a network of nodes with hyperlinks as connected edges, it turns out that information containing the importance or centrality of each site is contained with the dominant eigenvector. The power iteration is thus ideally suited for Google's needs.