

application of LINEAR ALGEBRA

# **Power Method To Find Dominant Eigen Value and Eigen Vector**

**NAME:RAJLAKSHMANAN C**

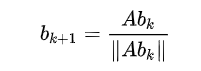
**UNIQUE ID:E0219054**

**WHAT IS POWER METHOD:**

In mathematics, **power iteration** (also known as the **power method**) is an eigenvalue algorithm given a diagonalizable matrix A {\displaystyle A} , the algorithm will produce a number λ {\displaystyle \lambda } , which is the greatest (in absolute value) eigenvalue of A {\displaystyle A} , and a nonzero vector v {\displaystyle v} , which is a corresponding eigenvector of λ {\displaystyle \lambda } , that is, A v = λ v {\displaystyle Av=\lambda v} . The algorithm is also known as the **Von Mises iteration**.[[1]](https://en.wikipedia.org/wiki/Power_iteration#cite_note-VonMises-1)

**METHOD:**

The power iteration algorithm starts with a vector b 0 {\displaystyle b\_{0}} , which may be an approximation to the dominant eigenvector or a random vector. The method is described by the recurrence relation



b k + 1 = A b k ‖ A b k ‖ {\displaystyle b\_{k+1}={\frac {Ab\_{k}}{\|Ab\_{k}\|}}}

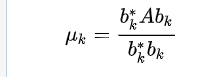
So, at every iteration, the vector b k {\displaystyle b\_{k}} is multiplied by the matrix A {\displaystyle A} and normalized.

If we assume A {\displaystyle A} has an eigenvalue that is strictly greater in magnitude than its other eigenvalues and the starting vector b 0 {\displaystyle b\_{0}} has a nonzero component in the direction of an eigenvector associated with the dominant eigenvalue, then a subsequence ( b k ) {\displaystyle \left(b\_{k}\right)} converges to an eigenvector associated with the dominant eigenvalue.

Without the two assumptions above, the sequence ( b k ) {\displaystyle \left(b\_{k}\right)} does not necessarily converge. In this sequence,

I:\Screenshot (219).pngb k = e i ϕ k v 1 + r k {\displaystyle b\_{k}=e^{i\phi \_{k}}v\_{1}+r\_{k}}

where v 1 {\displaystyle v\_{1}} is an eigenvector associated with the dominant eigenvalue, and ‖ r k ‖ → 0 {\displaystyle \|r\_{k}\|\rightarrow 0} . The presence of the term e i ϕ k {\displaystyle e^{i\phi \_{k}}} implies that ( b k ) {\displaystyle \left(b\_{k}\right)} does not converge unless e i ϕ k = 1 {\displaystyle e^{i\phi \_{k}}=1} . Under the two assumptions listed above, the sequence ( μ k ) {\displaystyle \left(\mu \_{k}\right)} defined by

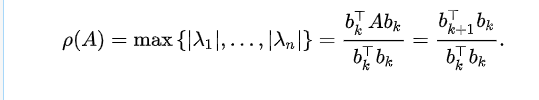
μ k = b k ∗ A b k b k ∗ b k {\displaystyle \mu \_{k}={\frac {b\_{k}^{\*}Ab\_{k}}{b\_{k}^{\*}b\_{k}}}}

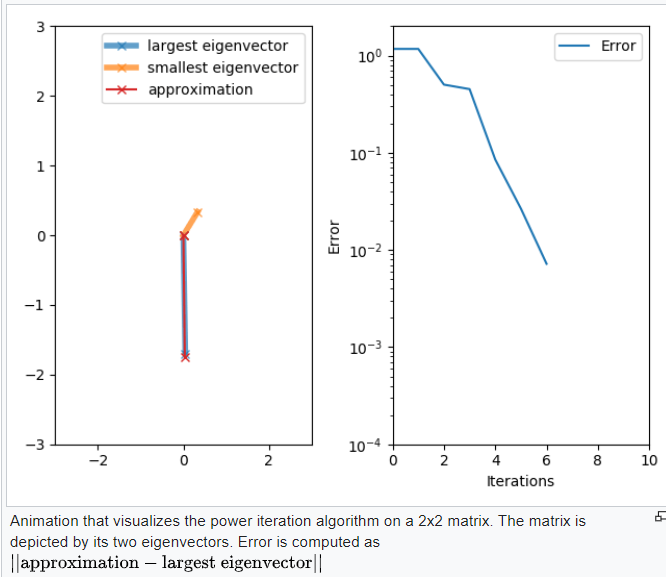
converges to the dominant eigenvalue

The vector b k {\displaystyle b\_{k}} to an associated eigenvector. Ideally, one should use the [Rayleigh quotient](https://en.wikipedia.org/wiki/Rayleigh_quotient) in order to get the associated eigenvalue.

This algorithm is used to calculate the *Google* [*PageRank*](https://en.wikipedia.org/wiki/PageRank).

The method can also be used to calculate the [spectral radius](https://en.wikipedia.org/wiki/Spectral_radius) (the eigenvalue with the largest magnitude, for a square matrix) by computing the Rayleigh quotient





## Algorithm for Power Method

**1. Start**

**2. Read Order of Matrix (n) and Tolerable Error (e)**

**3. Read Matrix A of Size n x n**

**4. Read Initial Guess Vector X of Size n x 1**

**5. Initialize: Lambda\_Old = 1**

**6. Multiply: X\_NEW = A \* X**

**7. Replace X by X\_NEW**

**8. Find Largest Element (Lamda\_New) by Magnitude from X\_NEW**

**9. Normalize or Divide X by Lamda\_New**

**10. Display Lamda\_New and X**

**11. If |Lambda\_Old - Lamda\_New| > e then**

**set Lambda\_Old = Lamda\_New and goto**

**step (6) otherwise goto step (12)**

**12. Stop**

# Power Method Pseudocode for Finding Dominant Eigen Value and Eigen Vector

**1. Start**

**2. Input:**

**a. Order of Matrix (n)**

**b. Tolerable Error (e)**

**3. Read Matrix (A):**

**For i = 1 to n**

**For j = 1 to n**

**Read Ai,j**

**Next j**

**Next i**

**4. Read Initial Guess Vector (X):**

**For i = 1 to n**

**Read Xi**

**Next i**

**5. Initialize: Lambda\_Old = 1**

**6. Multiplication (X\_NEW = A \* X):**

**For i = 1 to n**

**Temp = 0.0**

**For j = 1 to n**

**Temp = Temp + Ai,j \* Xj**

**Next j**

**X\_NEWi = Temp**

**Next i**

**7. Replace X by X\_NEW:**

**For i = 1 to n**

**Xi = X\_NEWi**

**Next i**

**8. Finding Largest:**

**Lambda\_New = |X1|**

**For i = 2 to n**

**If |Xi| > Lambda\_New**

**Lambda\_New = |Xi|**

**End If**

**Next i**

**9. Normalization:**

**For i = 1 to n**

**Xi = Xi/Lambda\_New**

**Next i**

**10. Display:**

**Print Lambda\_New**

**For i = 1 to n**

**Print Xi**

**Next i**

**11. Checking Accuracy:**

**If |Lambda\_New - Lambda\_Old| > e**

**Lambda\_Old = Lambda\_New**

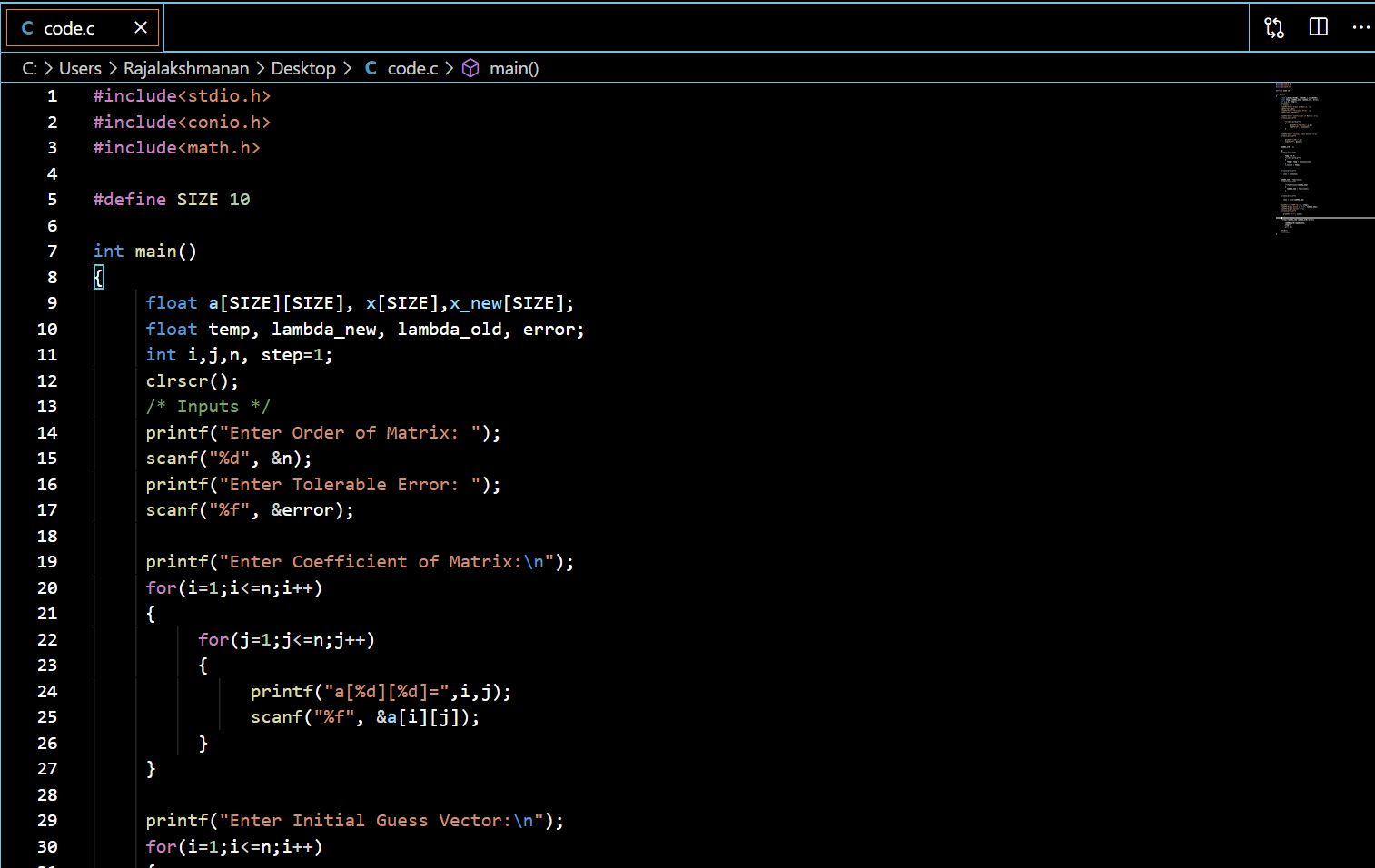
**Goto Step (6)**

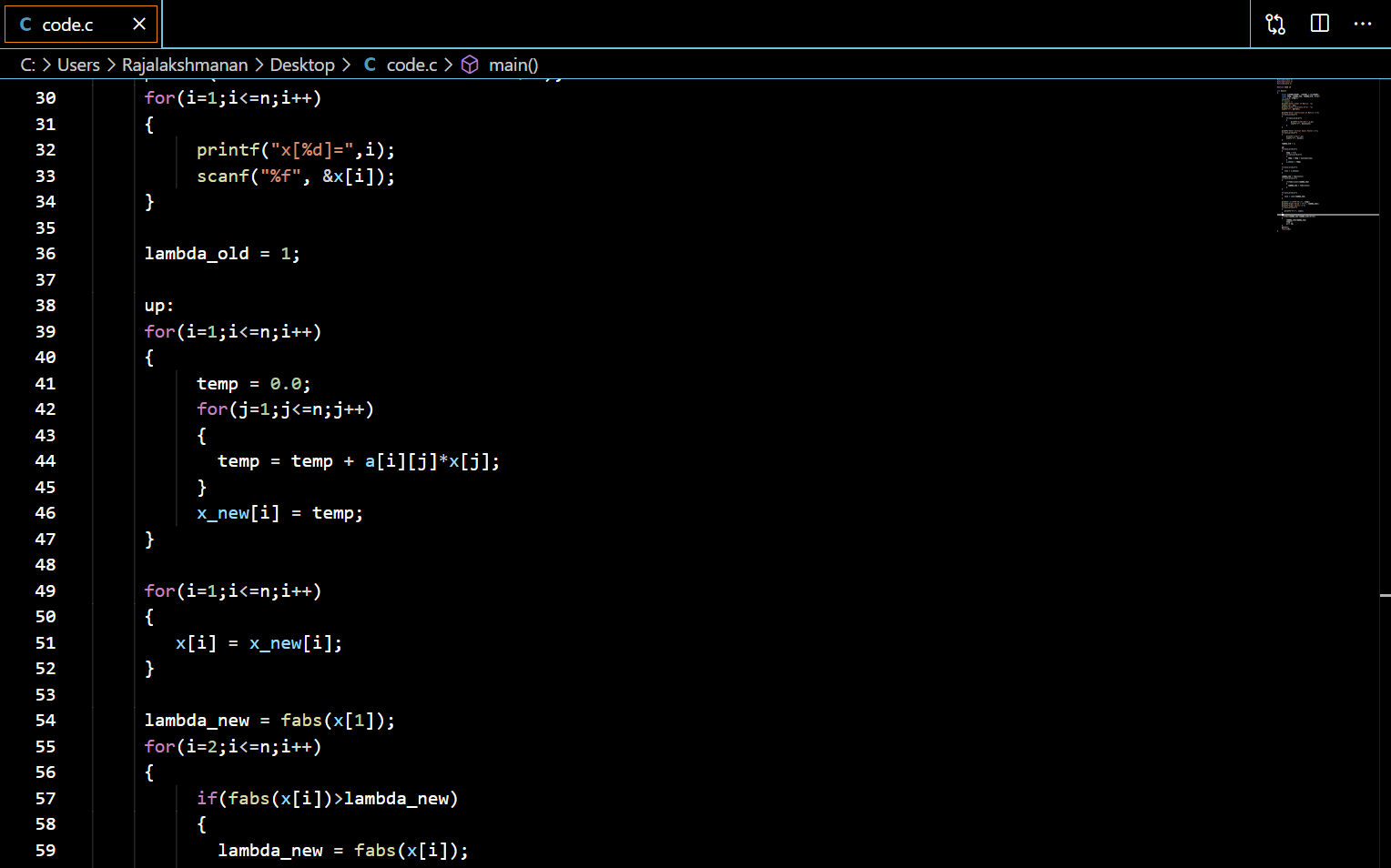
**End If**

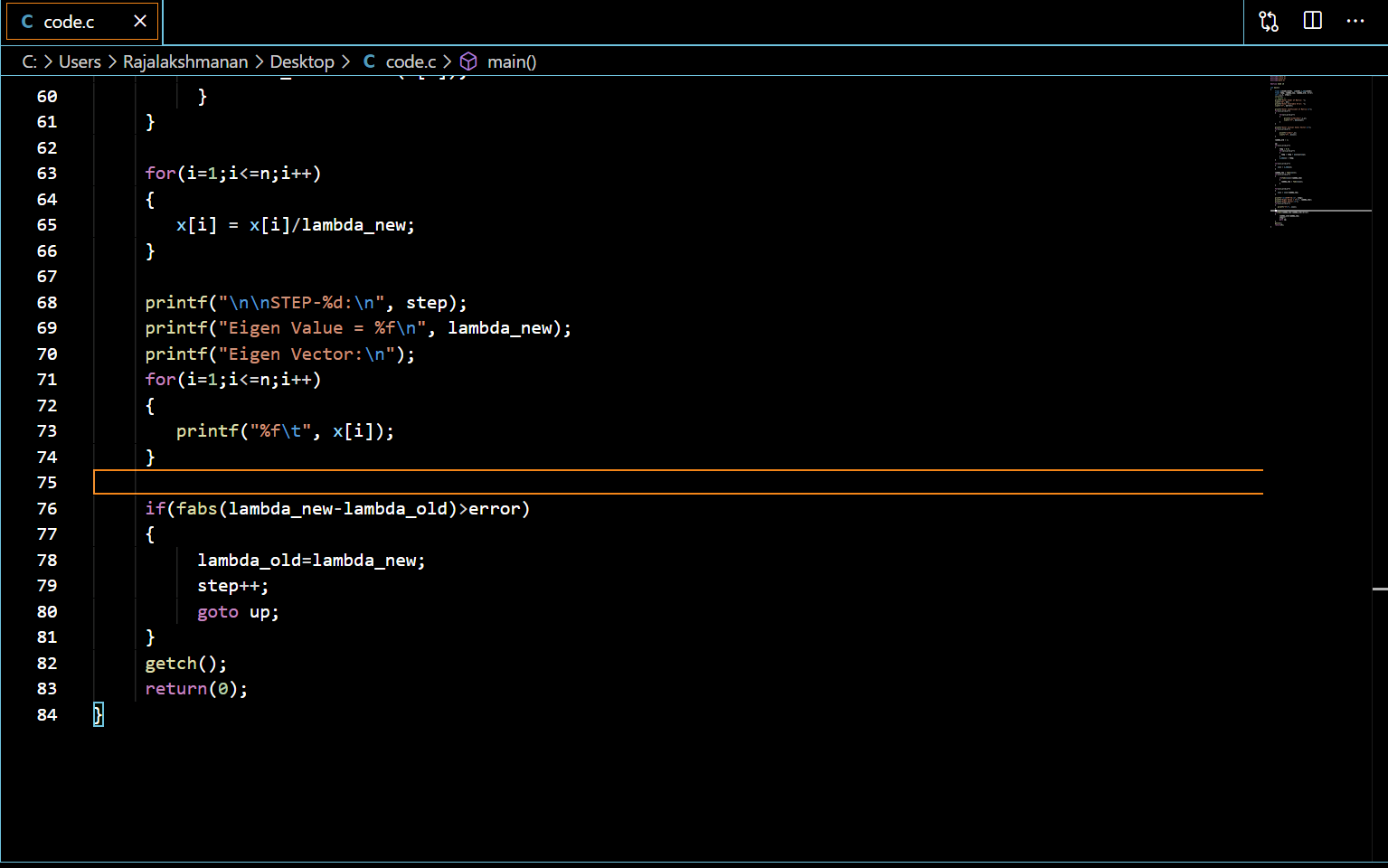
**12. Stop**

**Note: All array indexes are assumed to start from 1.**

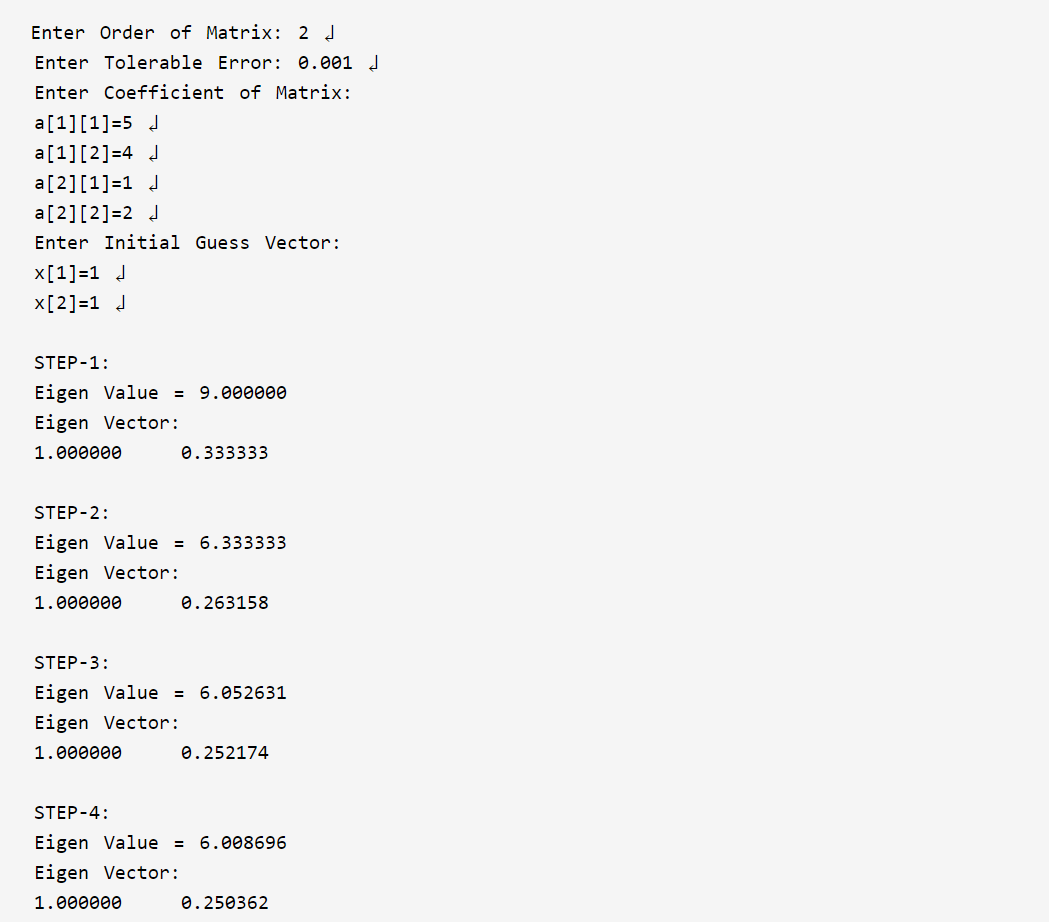
# Power Method Using C Programming for Finding Dominant Eigen Value and Eigen Vector

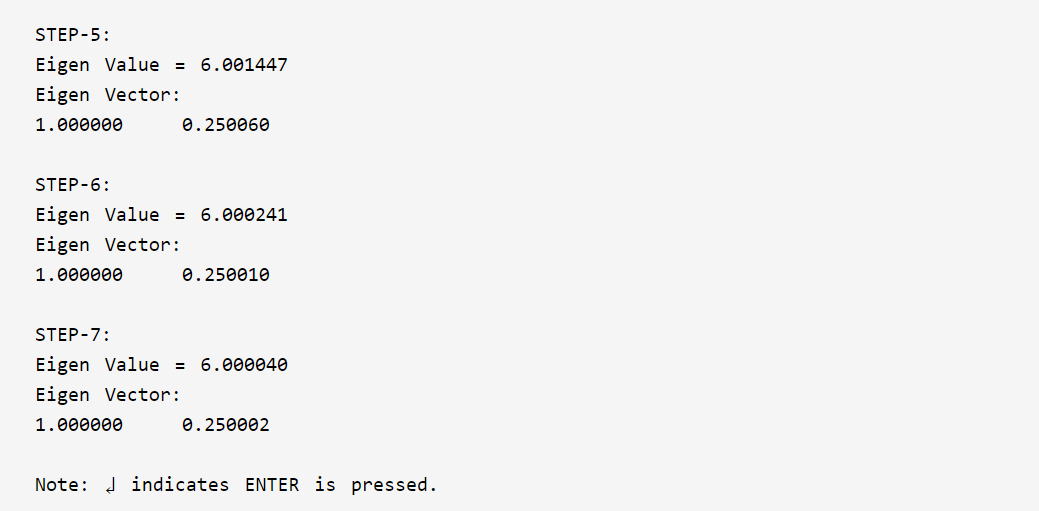




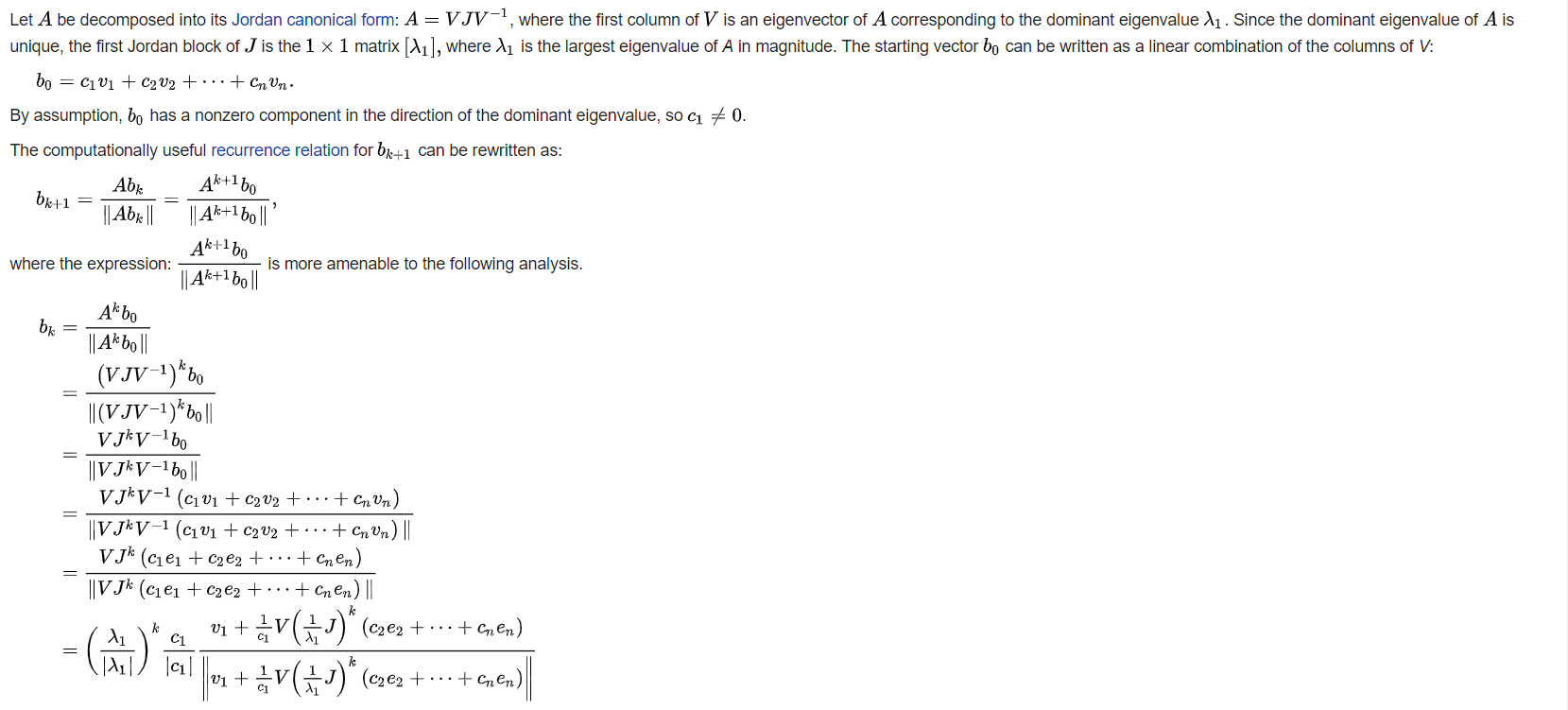


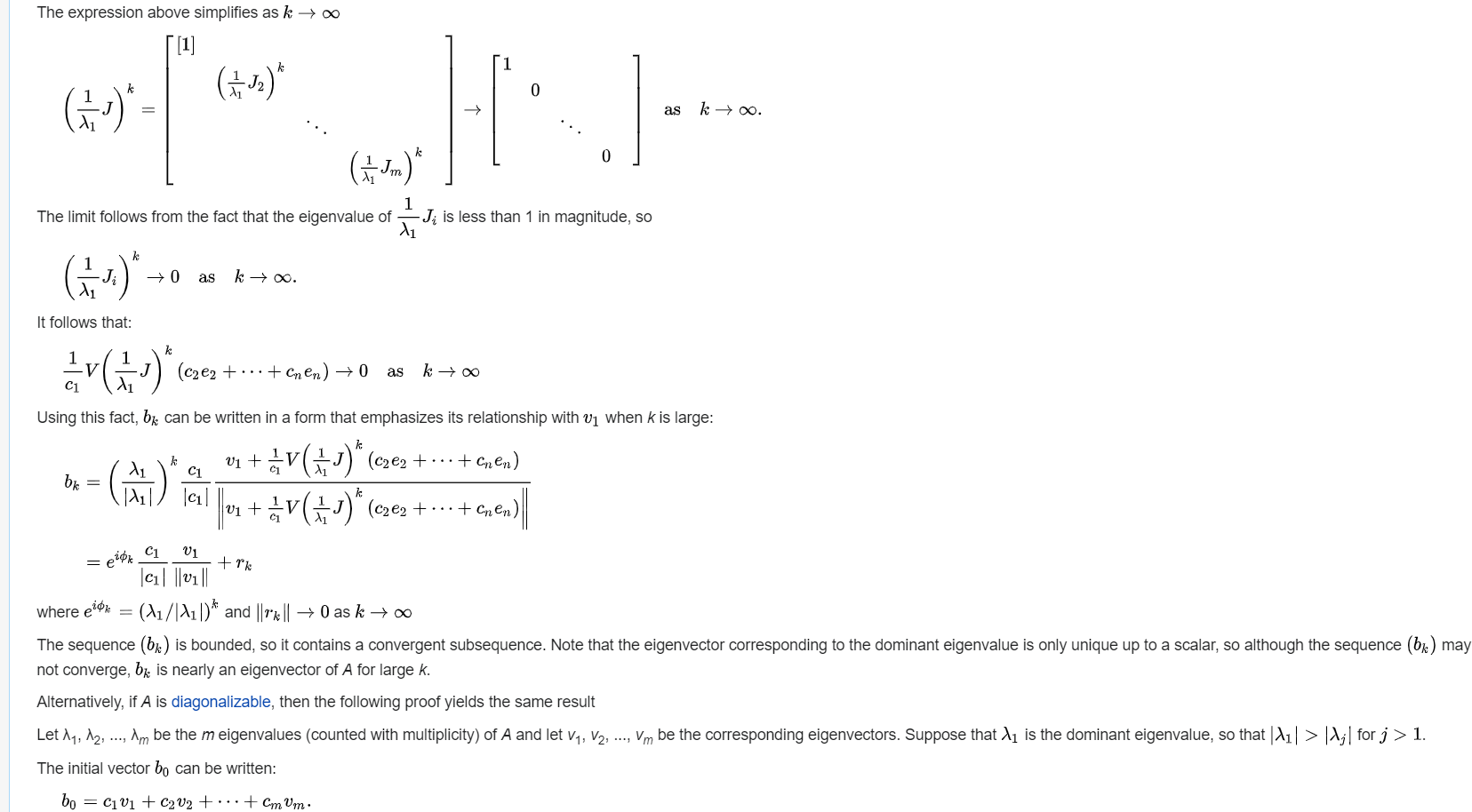
### Output : Power Method Using C Programming

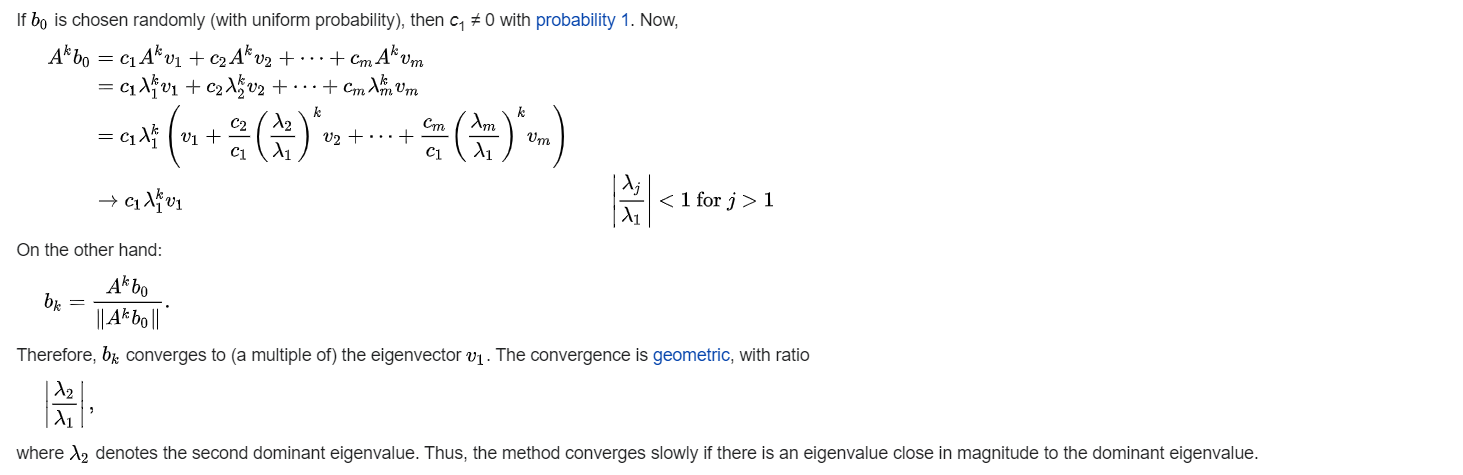




**Analysis**







**Applications:**

**Although the power iteration method approximates only one eigenvalue of a matrix, it remains useful for certain** [**computational problems**](https://en.wikipedia.org/wiki/Computational_problem)**. For instance,** [**Google**](https://en.wikipedia.org/wiki/Google) **uses it to calculate the** [**PageRank**](https://en.wikipedia.org/wiki/PageRank) **of documents in their search engine,**[**[2]**](https://en.wikipedia.org/wiki/Power_iteration#cite_note-2) **and** [**Twitter**](https://en.wikipedia.org/wiki/Twitter) **uses it to show users recommendations of whom to follow.**[**[3]**](https://en.wikipedia.org/wiki/Power_iteration#cite_note-twitterwtf-3) **The power iteration method is especially suitable for** [**sparse matrices**](https://en.wikipedia.org/wiki/Sparse_matrix)**, such as the web matrix, or as the** [**matrix-free method**](https://en.wikipedia.org/wiki/Matrix-free_methods) **that does not require storing the coefficient matrix A {\displaystyle A} explicitly, but can instead access a function evaluating matrix-vector products A x {\displaystyle Ax} . For non-symmetric matrices that are** [**well-conditioned**](https://en.wikipedia.org/wiki/Condition_number#matrices) **the power iteration method can outperform more complex** [**Arnoldi iteration**](https://en.wikipedia.org/wiki/Arnoldi_iteration)**. For symmetric matrices, the power iteration method is rarely used, since its convergence speed can be easily increased without sacrificing the small cost per iteration; see, e.g.,** [**Lanczos iteration**](https://en.wikipedia.org/wiki/Lanczos_iteration) **and** [**LOBPCG**](https://en.wikipedia.org/wiki/LOBPCG)**.**

**Some of the more advanced eigenvalue algorithms can be understood as variations of the power iteration. For instance, the** [**inverse iteration**](https://en.wikipedia.org/wiki/Inverse_iteration) **method applies power iteration to the matrix A − 1 {\displaystyle A^{-1}} . Other algorithms look at the whole subspace generated by the vectors b k {\displaystyle b\_{k}} . This subspace is known as the** [**Krylov subspace**](https://en.wikipedia.org/wiki/Krylov_subspace)**. It can be computed by** [**Arnoldi iteration**](https://en.wikipedia.org/wiki/Arnoldi_iteration) **or** [**Lanczos iteration**](https://en.wikipedia.org/wiki/Lanczos_iteration)**.**