12.664

Harsha-EE25BTECH11026

September 20,2025

Question

A real, invertible 3×3 matrix $\mathbf{M}$ has eigenvalues λ_i, $(i = 1, 2, 3)$ and the corresponding eigenvectors are $\mathbf{e_i}$, $(i = 1, 2, 3)$ respectively. Which one of the following is correct?

1. $\mathbf{Me_i} = \frac{1}{\lambda_i}\mathbf{e_i}$, for i=1,2,3
2. $\mathbf{M}^{-1}\mathbf{e_i} = \frac{1}{\lambda_i}\mathbf{e_i}$, for i=1,2,3
3. $\mathbf{M}^{-1}\mathbf{e_i} = \lambda_i\mathbf{e_i}$, for i=1,2,3
4. The eigenvalues of $\mathbf{M}$ and $\mathbf{M}^{-1}$ are not related.

Theoretical Solution

According to the definition of eigen-vector,

$$\mathbf{M}\mathbf{e_i} = \lambda_i \mathbf{e_i} \tag{1}$$

Pre-multiplying $\mathbf{M}^{-1}$ on both sides,

$$\therefore \left(\mathbf{M}^{-1}\mathbf{M}\right)\mathbf{e_i} = \mathbf{M}^{-1}\lambda_i \mathbf{e_i} \tag{2}$$

$$\implies \mathbf{e_i} = \lambda_i \mathbf{M}^{-1}\mathbf{e_i} \tag{3}$$

$$\therefore \mathbf{M}^{-1}\mathbf{e_i} = \frac{1}{\lambda_i}\mathbf{e_i} \tag{4}$$

C Code -Verifying the relation between the eigenvalues of a matrix and its inverse

```c
#include <stdio.h>
#include <math.h>

#define N 10

int inverse(int n, double A[N][N], double Inv[N][N]) {
    double aug[N][2*N];

    // Form augmented matrix [A|I]
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            aug[i][j] = A[i][j];
        }
        for (int j = 0; j < n; j++) {
            aug[i][j+n] = (i==j) ? 1.0 : 0.0;
        }
    }
```

C Code -Verifying the relation between the eigenvalues of a matrix and its inverse

```
    for (int i = 0; i < n; i++) {
        double pivot = aug[i][i];
        if (fabs(pivot) < 1e-12) {
            return 0; // singular
        }
        // Normalize row
        for (int j = 0; j < 2*n; j++) {
            aug[i][j] /= pivot;
        }
        // Eliminate other rows
        for (int k = 0; k < n; k++) {
            if (k != i) {
                double factor = aug[k][i];
                for (int j = 0; j < 2*n; j++) {
                    aug[k][j] -= factor * aug[i][j];
                }
            }}}
```

C Code -Verifying the relation between the eigenvalues of a matrix and its inverse

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            Inv[i][j] = aug[i][j+n];
        }
    }

    return 1;
}

void qr_iteration(int n, double A[N][N], double eigvals[N], int
    max_iter, double tol) {
    double Q[N][N], R[N][N], Ak[N][N];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            Ak[i][j] = A[i][j];
```

C Code -Verifying the relation between the eigenvalues of a matrix and its inverse

```c
for (int iter = 0; iter < max_iter; iter++) {
        // Gram-Schmidt to compute QR
        for (int j = 0; j < n; j++) {
            for (int i = 0; i < n; i++) Q[i][j] = Ak[i][j];
            for (int k = 0; k < j; k++) {
                double dot = 0;
                for (int i = 0; i < n; i++) dot += Q[i][k] * Ak[i
                    ][j];
                for (int i = 0; i < n; i++) Q[i][j] -= dot * Q[i][
                    k];
            }
            double norm = 0;
            for (int i = 0; i < n; i++) norm += Q[i][j]*Q[i][j];
            norm = sqrt(norm);
            for (int i = 0; i < n; i++) Q[i][j] /= norm;
        }
```

C Code -Verifying the relation between the eigenvalues of a matrix and its inverse

```c
  // R = Q^T * A
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                R[i][j] = 0;
                for (int k = 0; k < n; k++) R[i][j] += Q[k][i] *
                    Ak[k][j];
            }
        // Ak = R * Q
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                Ak[i][j] = 0;
                for (int k = 0; k < n; k++) Ak[i][j] += R[i][k] *
                    Q[k][j];
            }
    }
    for (int i = 0; i < n; i++) eigvals[i] = Ak[i][i];
}
```

Python+C code

```python
import ctypes
import numpy as np

lib = ctypes.CDLL("./libmatrix_ops.so")

N = 10
MAX_ITER = 1000
TOL = 1e-9
# Define argument types
lib.inverse.argtypes = [ctypes.c_int,
                        (ctypes.c_double * N * N),
                        (ctypes.c_double * N * N)]
lib.inverse.restype = ctypes.c_int
lib.qr_iteration.argtypes = [ctypes.c_int,(ctypes.c_double * N *
    N),
                (ctypes.c_double * N),ctypes.c_int,ctypes.c_double
                    ]
```

Python+C code

```python
def matrix_inverse(A: np.ndarray):
    n = A.shape[0]
    A_c = (ctypes.c_double * N * N)()
    Inv_c = (ctypes.c_double * N * N)()

    for i in range(n):
        for j in range(n):
            A_c[i][j] = A[i, j]

    success = lib.inverse(n, A_c, Inv_c)
    if not success:
        raise ValueError("Matrix is singular")
```

Python+C code

```
Inv = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            Inv[i, j] = Inv_c[i][j]

    return Inv

# Wrapper: Eigenvalues
def eigenvalues(A: np.ndarray):
    n = A.shape[0]
    A_c = (ctypes.c_double * N * N)()
    eig_c = (ctypes.c_double * N)()

    for i in range(n):
        for j in range(n):
            A_c[i][j] = A[i, j]
    lib.qr_iteration(n, A_c, eig_c, MAX_ITER, TOL)
    return np.round(np.array([eig_c[i] for i in range(n)]), 3)
```

Python+C code

```
# Example
if __name__ == "__main__":
    A = np.array([[4.0, 2.0, 1.0],
                  [1.0, 3.0, 2.0],
                  [0.0, 5.0, 6.0]])
    B=matrix_inverse(A)
    print("Eigenvalues of A:",eigenvalues(A))
    print("Eigenvalues of B:",eigenvalues(B))
```

Python code

```python
import numpy as np

#Example matrix
A=np.matrix([[1,3,4],[5,4,9],[2,7,6]])

B=np.linalg.inv(A)

eigvals=np.linalg.eigvals(A)
print("The eigen values of matrix A:",np.round(eigvals,3))
eigvals_inv=np.linalg.eigvals(B)
print(r"The eigen values of matrix B:",np.round(eigvals_inv,3))
```