12.368

EE25BTECH11043 - Nishid Khandagre

October 8, 2025

Question

If the nullity of the matrix $\begin{pmatrix} k & 1 & 2 \\ 1 & -1 & -2 \\ 1 & 1 & 4 \end{pmatrix}$ is 1, then the value of k is ?

Solution

Given matrix:

$$\mathbf{A} = \begin{pmatrix} k & 1 & 2 \\ 1 & -1 & -2 \\ 1 & 1 & 4 \end{pmatrix} \tag{1}$$

Nullity $= 1$ for a 3×3 matrix means $\text{rank}(A) = 2$.

Solution

Swap R_1 and R_2:

$$\begin{pmatrix} 1 & -1 & -2 \\ k & 1 & 2 \\ 1 & 1 & 4 \end{pmatrix} \tag{2}$$

Apply $R_2 \to R_2 - kR_1$ and $R_3 \to R_3 - R_1$:

$$\begin{pmatrix} 1 & -1 & -2 \\ 0 & 1+k & 2+2k \\ 0 & 2 & 6 \end{pmatrix} \tag{3}$$

Solution

For the rank to be 2, the last two rows must be linearly dependent. Thus, there exists a scalar t such that:

$$1 + k = 2t \tag{4}$$

$$2 + 2k = 6t \tag{5}$$

Divide the equations

$$2 + 2k = 3(1 + k) \tag{6}$$

$$2 + 2k = 3 + 3k \tag{7}$$

$$k = -1 \tag{8}$$

Thus, the value of k is -1.

C Code

```c
#include <stdio.h>
// Function to calculate the determinant of a 3x3 matrix
// The matrix is passed as a flat array of 9 doubles for
    simplicity in ctypes.
//order: [a11, a12, a13, a21, a22, a23, a31, a32, a33]
double calculate_determinant(double* matrix_elements) {
    double a = matrix_elements[0]; double b = matrix_elements[1];
         double c = matrix_elements[2];
    double d = matrix_elements[3]; double e = matrix_elements[4];
         double f = matrix_elements[5];
    double g = matrix_elements[6]; double h = matrix_elements[7];
         double i = matrix_elements[8];

    // Formula for 3x3 determinant:
    // a(ei - fh) - b(di - fg) + c(dh - eg)
    return a * (e * i - f * h) - b * (d * i - f * g) + c * (d * h
         - e * g);
}
```

Python Code (using C shared library)

```
import ctypes
import numpy as np
# Load the shared library
lib_matrix = ctypes.CDLL(./code22.so)
# Define the argument types and return type for the C function
lib_matrix.calculate_determinant.argtypes = [
    ctypes.POINTER(ctypes.c_double)
]
lib_matrix.calculate_determinant.restype = ctypes.c_double
def get_determinant_from_c(k_value):
#Constructs the matrix for a given k_value, flattens it,
#and calls the C function to get its determinant.

    matrix_elements_flat = (ctypes.c_double * 9)(
        k_value, 1.0, 2.0,
        1.0, -1.0, -2.0,
        1.0, 1.0, 4.0
    )
```

Python Code (using C shared library)

```
determinant = lib_matrix.calculate_determinant(
    matrix_elements_flat)
    return determinant
def solve_for_k_with_c_determinant():
#Finds the value of k such that the nullity of the matrix is 1.
#This implies the determinant of the matrix is 0.
    print(Solving for 'k' in the matrix problem where nullity is
        1 (determinant = 0).\n)
    k_solution_algebraic = -1.0
    print(f1. Algebraic Solution: From det = -2k - 2 = 0, we find
         k = {k_solution_algebraic:.0f})

    print(\n2. Verification using the C function for k = -1:)
    verified_determinant = get_determinant_from_c(
        k_solution_algebraic)
    print(f For k = {k_solution_algebraic:.0f}, the determinant (
        from C code) is: {verified_determinant:.6f})
```

Python Code (using C shared library)

```
    if abs(verified_determinant) < 1e-9:
        print( The determinant is approximately zero, confirming k = -1 is correct.)
    else:
        print( The determinant is not zero. There might be an issue with calculation or assumption.)

print(f\nTherefore, the value of k for which the nullity of the matrix is 1 is: {k_solution_algebraic:.0f})

print(\n--- Visualizing the matrix with k = -1 ---)
final_matrix_k = -1
final_matrix = np.array([
    [final_matrix_k, 1, 2],
    [1, -1, -2],
    [1, 1, 4]
])
```

Python Code (using C shared library)

```
    print(final_matrix)
    print(f"Numpy's determinant for this matrix: {np.linalg.det(
        final_matrix):.6f}")
    print(f"Numpy's rank for this matrix: {np.linalg.matrix_rank(
        final_matrix)}")
    print(f"Numpy's nullity (columns - rank): {final_matrix.shape
        [1] - np.linalg.matrix_rank(final_matrix)}")

# Run the solver
solve_for_k_with_c_determinant()
```

Python Code (Pure Python)

```
import numpy as np

def calculate_determinant_pure_python(k_val):

    Calculates the determinant of the given 3x3 matrix using
        NumPy.
    The matrix structure is:
    [ k 1 2 ]
    [ 1 -1 -2 ]
    [ 1 1 4 ]

    matrix = np.array([
        [k_val, 1, 2],
        [1, -1, -2],
        [1, 1, 4]
    ])
    return np.linalg.det(matrix)
```

Python Code (Pure Python)

```
def solve_matrix_problem_pure_python():

    For a 3x3 matrix, nullity 1 means rank is 2.
    For rank to be 2 (and not 3), the determinant must be 0.

    k_solution = -1.0
    print(f1. Manual Algebraic Derivation:)
    print(f The determinant of the matrix is -2k - 2.)
    print(f For nullity to be 1, the determinant must be 0.)
    print(f Setting -2k - 2 = 0 gives k = {k_solution:.0f}.\n)

    print(f2. Verification using NumPy's determinant function:)
    matrix_with_k = np.array([
        [k_solution, 1, 2],
        [1, -1, -2],
        [1, 1, 4]
    ])
```

Python Code (Pure Python)

```
    print( Matrix with k = -1:)
    print(matrix_with_k)

    det_verified = np.linalg.det(matrix_with_k)
    print(f Determinant (calculated by NumPy): {det_verified:.6f})

    if abs(det_verified) < 1e-9:
        print( The determinant is approximately zero, confirming k = -1 is correct for rank < 3.)
    else:
        print( Error: Determinant is not zero as expected.)

    rank = np.linalg.matrix_rank(matrix_with_k)
    nullity = matrix_with_k.shape[1] - rank
```

Python Code (Pure Python)

```
    print(f Rank of the matrix: {rank})
    print(f Nullity of the matrix (columns - rank): {nullity}\n)

    if nullity == 1:
        print(fConclusion: The value of k that results in a nullity of 1 is: {k_solution:.0f})
    else:
        print(Conclusion: The calculated k did not result in a nullity of 1.)

# Run the solver
solve_matrix_problem_pure_python()
```