```matlab
function F = matrix_inversion_lemma(A_inv, B, C, D)
    % This function implements the Matrix Inversion Lemma:
    % (A + BCD)^(-1) = A^(-1) - A^(-1)B(C^(-1) + DA^(-1)B)^(-1)DA^(-1)

    % Inputs:
    % A_inv - Inverse of matrix A (A^(-1)) [Assumed given]
    % B, C, D - Matrices involved in the lemma

    % Output:
    % F - The result of the Matrix Inversion Lemma

    % Step 0: Check dimensions of A_inv, B, and D
    [m_A, n_A] = size(A_inv);
    [m_B, n_B] = size(B);
    [m_D, n_D] = size(D);

    if n_A ~= m_A
        error('A_inv must be a square matrix.');
    end

    if m_A ~= m_B
        error('The number of rows in B must match the number of rows in A_inv.');
    end

    if n_B ~= m_D
        error('The number of columns in B must match the number of rows in D.');
    end

    if n_D ~= m_A
        error('The number of columns in D must match the number of columns in A_inv.');
    end

    % Step 1: Compute the inverse of C
    if isscalar(C)
        C_inv = 1 / C;  % If C is a scalar, simply take the reciprocal
    else
        C_inv = inv(C);  % Otherwise, compute the matrix inverse
    end

    % Step 2: Compute the term (C^(-1) + DA^(-1)B)
    intermediate_term = C_inv + D * A_inv * B;

    % Step 3: Compute the inverse of the intermediate term
    intermediate_term_inv = inv(intermediate_term);

    % Step 4: Apply the matrix inversion lemma formula
    F = A_inv - A_inv * B * intermediate_term_inv * D * A_inv;
end

% Test the function with given matrices
A_inv = diag([2, 1, 1, 2, 1]);  % A_inv is the inverse of A
B = [3; 0; 2; 0; 1];
C = 0.25;  % C is a scalar
D = transpose(B);
```

```
F = matrix_inversion_lemma(A_inv, B, C, D);
disp(F);
```

```
    0.6667         0   -0.4444         0   -0.2222
         0    1.0000         0         0         0
   -0.4444         0    0.8519         0   -0.0741
         0         0         0    2.0000         0
   -0.2222         0   -0.0741         0    0.9630
```

Published with MATLAB® R2024a