An implementation of the Gauss-Jordan method

Derek W. Harrison

July 1, 2024

Introduction

The Gauss-Jordan method is implemented to compute the inverse of a matrix in $O\left(n^3\right)$ time. The pseudocode of the method is given in the next section. The procedure GAUSS-JORDAN(A, n, C) performs the inversion and takes as input a square matrix A and the size of the input matrix n and stores the inverse of A in C. Note the procedure does not check if the input matrix is singular.

Algorithm

```
function GAUSS-JORDAN(A, n, C)
    INIT-MAT(C, n)
    //Convert to row echelon form
    for c = 0; c < n; c = c + 1 do
        if A[c][c] == 0 then
            r = FIND(A, c, n)
            SWAP(A, r, c, n)
            SWAP(C, r, c, n)
        end if
        //Normalize rows
        for j = c + 1; j < n; j = j + 1 do
            A[c][j] = A[c][j]/A[c][c]
        end for
        for j = 0; j < n; j = j + 1 do
            C[c][j] = C[c][j]/A[c][c]
        end for
        A[c][c] = 1.0
        //Delete elements in rows below
        for r = c + 1; r < n; r = r + 1 do
            if A[r][c] ≠ 0 then
                for j = c + 1; j < n; j = j + 1 do
                    A[r][j] = −A[r][c] · A[c][j] + A[r][j]
                end for
                for j = 0; j < n; j = j + 1 do
                    C[r][j] = −A[r][c] · C[c][j] + C[r][j]
                end for
                A[r][c] = 0
            end if
        end for
    end for
    //Backtrack to convert to reduced row echelon form
    for c = n − 1; c > 0; c = c − 1 do
        for r = c − 1; r > −1; r = r − 1 do
            if A[r][c] ≠ 0 then
                for j = 0; j < n; j = j + 1 do
                    C[r][j] = −A[r][c] · C[c][j] + C[r][j]
                end for
                A[r][c] = 0
            end if
        end for
    end for
end function
```

```
function FIND(A, c, n)
    for r = c + 1; r < n; r = r + 1 do
        if A[r][c] ≠ 0 then
            return r
        end if
    end for
    return −1
end function
```

```
function INIT-MAT(A, n)
    for i = 0; i < n; i = i + 1 do
        for j = 0; j < n; j = j + 1 do
            if i == j then
                A[i][j] = 1
            else
                A[i][j] = 0
            end if
        end for
    end for
end function
```

```
function SWAP(M, r, c, n)
    Let row be an array of size n
    for i = 0; i < n; i = i + 1 do
        row[i] = M[r][i]
        M[r][i] = M[c][i]
    end for
    for i = 0; i < n; i = i + 1 do
        M[c][i] = row[i]
    end for
end function
```