

An implementation of the Gauss-Jordan method

Derek W. Harrison

May 9, 2022

Introduction

The Gauss-Jordan method is implemented with row sorting instead of row swapping to compute the inverse of a matrix in $O(n^3)$ time. The pseudocode of the method is given in the next section. The procedure $\text{GAUSS-JORDAN}(A, n, C)$ 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 . The procedure assumes the input matrix is non-singular. Merge sort is used to sort the rows of the matrices according to the number of leading zeros. It is the underlying algorithm of the $\text{SORT-MAT}(M, n, C)$ procedure used to sort the matrices.

Algorithm

```
function GAUSS-JORDAN( $A, n, C$ )
  INIT-MAT( $C, n$ )
  let  $M[0..n-1]$  be a new array
  //Convert to reduced row echelon form
for  $c = 0; c < n; c = c + 1$  do
    if  $A[c][c] == 0$  then
      GET-ORDER( $A, n, M$ )
      SORT-MAT( $M, n, A$ )
      SORT-MAT( $M, n, C$ )
    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  $A[r][c] \neq 0$  then
      for  $j = c + 1; j < n; j = j + 1$  do
         $A[r][j] = -A[r][c] \cdot A[c][j] + A[r][j]$ 
      end for
      for  $j = 0; j < n; j = j + 1$  do
         $C[r][j] = -A[r][c] \cdot C[c][j] + C[r][j]$ 
      end for
       $A[r][c] = 0$ 
    end if
  end for
end for
  //Backtrace to complete conversion 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  $A[r][c] \neq 0$  then
      for  $j = 0; j < n; j = j + 1$  do
         $C[r][j] = -A[r][c] \cdot C[c][j] + C[r][j]$ 
      end for
       $A[r][c] = 0$ 
    end if
  end for
end for
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 GET-ORDER( $A, n, M$ )
  for  $i = 0; i < n; i = i + 1$  do
     $c = 0$ 
    while  $A[i][c] == 0$  and  $c < n$  do
       $c = c + 1$ 
    end while
     $M[i] = c$ 
  end for
end function

```

```

function SORT-MAT( $M, n, A$ )
  Sort rows in  $A$  according to order array  $M$  using merge-sort.
end function

```