Gauss-Jordan Elimination

Stephen D. Butalla & Valerie Kobzarenko

August 26, 2019

CSE-5400
Florida Institute of Technology

Gauss-Jordan Elimination: Uses, Advantages, and Deficiencies

- Uses [1]
  - Solving sets of linear equations:

$$a_{00}x_0 + a_{01}x_1 + \cdots a_{0m}x_n = y_1$$
$$\vdots$$
$$a_{n0}x_0 + a_{n1}x_1 + \cdots a_{nm}x_n = y_n$$

    or

$$\mathbb{A}\mathbb{X} = \mathbb{Y}$$

    where $\mathbb{A}$ is an $N \times M$ matrix of coefficients, $\mathbb{X}$ is an $N \times 1$ column vector, and $\mathbb{Y}$ is an $N \times 1$ column vector
  - Finding the inverse of a matrix. I.e.,

$$\mathbb{A}^{-1}, \text{ such that } \mathbb{A} \cdot \mathbb{A}^{-1} = \mathbb{I}$$

- Advantages [1]
  - As efficient as other methods for finding an inverse matrix
  - Straight-forward and easy to use
- Deficiencies [1]
  - Three times slower when solving linear equations than the LU decomposition method
  - Values on the right hand side (RHS) of the set of equations must be manipulated and stored simltaneously
  - Inherently unstable with roundoff error when using the no-pivot method (the no-pivot method should never be used!)

Brief Review of Gauss-Jordan Elimination

- Brief review of how Gauss-Jordan method works for solving a system of linear equations. We take our coefficient matrix and adjoin the RHS vector:

$$\left[\begin{array}{ccc|c} a_{00} & a_{01} & a_{02} & y_0 \\ a_{10} & a_{11} & a_{12} & y_1 \\ a_{30} & a_{31} & a_{32} & y_2 \end{array}\right]$$

- We then perform row operations until we have the matrix in row-echelon form:

$$\left[\begin{array}{ccc|c} 1 & \tilde{a}_{01} & a_{02} & \tilde{y}_0 \\ 0 & 1 & \tilde{a}_{12} & \tilde{y}_1 \\ 0 & 0 & 1 & \tilde{y}_2 \end{array}\right]$$

  and use back-substitution to solve for x_0, x_1, and x_2.

- Reduced row-echelon form is preferred:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & \tilde{y}_0^* \\ 0 & 1 & 0 & \tilde{y}_1^* \\ 0 & 0 & 1 & \tilde{y}_2^* \end{array}\right]$$

  and is computationally easy using the algorithm.

Brief Review of Gauss-Jordan Elimination

- To find the inverse of a matrix, we adjoin the identity matrix $\mathbb{I}$ to the RHS:

$$\left[\begin{array}{ccc|ccc} a_{00} & a_{01} & a_{02} & 1 & 0 & 0 \\ a_{10} & a_{11} & a_{12} & 0 & 1 & 0 \\ a_{30} & a_{31} & a_{32} & 0 & 0 & 1 \end{array}\right]$$

- We then perform row operations until the LHS matrix $\equiv \mathbb{I}$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \tilde{a}_{00} & \tilde{a}_{01} & \tilde{a}_{02} \\ 0 & 1 & 0 & \tilde{a}_{10} & \tilde{a}_{11} & \tilde{a}_{12} \\ 0 & 0 & 1 & \tilde{a}_{30} & \tilde{a}_{31} & \tilde{a}_{32} \end{array}\right]$$

- The RHS is now our inverse matrix, $\mathbb{A}^{-1}$
- Caveat: if a row on the LHS contains all zeros, an inverse matrix does not exist

Brief Review of Gauss-Jordan Elimination

- Two methods can be used with the Gauss-Jordan prescription for solving linear equations/finding the inverse of a matrix: with and without pivoting
- The *pivot* or *pivot element* is the diagonal element we will divide the current row by (e.g., $a_{00}, a_{11}, a_{22}, ...$) [1]
- Gauss-Jordan Prescription
  - Start on row (R) 0, column (C) 0. Divide R0 by a_{00}
  - R0 is subtracted from other rows (e.g., R1, R2, ...). C0 of $\mathbb{A} \equiv$ R0 of $\mathbb{I}$
  - Divide R1 by a_{11}.
  - RHS of R1 subtracted from other rows (R0, R2, ...)
  - Repeat for remaining rows
- In this procedure, no rows or columns were swapped, which makes this method no pivot
- Problem: dividing an element of the current row by a zero in the diagonal of the row

Gauss-Jordan With Pivoting

- Pivoting allows us to strategically place the rows/columns so that getting a 1 on the diagonal is easier
- Partial pivoting: swapping a row/rows (this is easier than full pivoting)
- Full pivoting: swapping columns and rows (need to keep close track of column swapping)
- When pivoting, we need to make sure we don't disturb part of the identity matrix that we have in correct form
- Rule of thumb for selecting a good pivot: pick the highest magnitude element on the diagonal

Implementation of Gauss-Jordan With Pivoting

- Declare `int` arrays (in [1], we use the `VecInt` defined type, included in `nr3.h`) to keep track of the row and column swapping
- A main loop controls the reduction to reduced row-echelon form
- A nested loop looks for the pivot; once found, it swaps rows so the "ideal" pivot is in the correct location
- Divide the current row by the pivot then reduce the rows
- Repeat as necessary until the identity is produced
- Using the "tracking" arrays, restore the columns to the original configurations
- The resulting output will be the solutions to the linear equations

References

[1] W. Press, S. Teukolsky, W. Vetterling, & B. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 2007, Cambridge University Press.