Gaussian Elimination

MATLAB Implementation

Tamas Kis | kis@stanford.edu

TAMAS KIS
https://github.com/tamaskis

Contents

gaussian_elimination

Solves the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ using Gaussian elimination with partial pivoting.

Syntax

```
x = gaussian_elimination(A,b)
```

Description

`x = gaussian_elimination(A,b)` solves the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ for $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$.

Examples

Example 1

Solve the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, where

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 5 \\ 1 & 1 & -3 \\ 2 & 4 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 10 \\ -2 \\ 1 \end{bmatrix}$$

■ ***SOLUTION***

Entering $\mathbf{A}$ and $\mathbf{b}$ into MATLAB,

```
% defines matrix A
A = [2,-1, 5;
     1, 1,-3;
     2, 4, 1];

% defines vector b
b = [10;
     -2;
      1];
```

To solve the linear system for $\mathbf{x}$,

```
x = gaussian_elimination(A,b)
```

This yields the result

```
x =

     2
    -1
     1
```

Links

MATLAB® Central's File Exchange:
https://www.mathworks.com/matlabcentral/fileexchange/89306-gaussian-elimination-gaussian_elimination

GitHub®:
https://github.com/tamaskis/gaussian_elimination-MATLAB

Gaussian Elimination

Gaussian elimination can be used to solve the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{n\times 1}$. In Algorithms 1 and 2 below, we will be referring to the rows and columns of $\mathbf{A}$ as well as to the elements of $\mathbf{x}$ at different points. Here are the conventions we use:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_j & \dots & \mathbf{a}_{n-1} & \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \vdots \\ \bar{a}_i \\ \vdots \\ \bar{a}_{n-1} \\ \bar{a}_n \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

Algorithms 1 and 2[1] below implement Gaussian elimination with partial pivoting and are adapted from Algorithm 6.2 in [1, pp. 374–375]. Note that ε is used to denote the machine epsilon.

[1] Note that together, these algorithms represent a single algorithm. However, this single algorithm is split into two due to its length and the limitations of typesetting algorithms in LaTeX.

Algorithm 1: Gaussian elimination with partial pivoting (part 1).

Given: $\mathbf{A}, \mathbf{x}, \mathbf{b}$

// determines n (where $\mathbf{A} \in \mathbb{R}^{n \times n}$)

$n = \text{size}(\mathbf{A}, 1)$

// redefines A by augmenting the original A with $\mathbf{b}$

$\mathbf{A} = [\mathbf{A}\ \mathbf{b}]$

// initializes boolean variable to keep track if matrix is singular

$\text{singular} = \text{false}$

// elimination process

for $i = 1$ **to** $n - 1$ **do**

    // determines pivot row

    Define the vector $\mathbf{p} = (i, i+1, ..., n-1, n)^T$.

    Remove the elements p_k of $\mathbf{p}$ where $A_{k,i} = 0$.

    $p = \min(\mathbf{p})$

    // exit the loop if all possible pivots in the i^{th} column are zero (to machine precision) because that would result in a singular matrix

    if $\max |\mathbf{a}_i| \leq \varepsilon$ **then**

        $\text{singular} = \text{true}$

        Exit loop – the matrix is singular and no unique solution exists.

    end

    // switches pivot row with the i^{th} row if $p \neq i$

    if $p \neq i$ **then**

        Store the i^{th} row of $\mathbf{A}$ as $\bar{a}_i$.

        Store the p^{th} row of $\mathbf{A}$ as $\bar{a}_p$.

        Set the i^{th} row of $\mathbf{A}$ to $\bar{a}_p$.

        Set the p^{th} row of $\mathbf{A}$ to $\bar{a}_i$.

    end

    // elementary row operation

    for $j = i + 1$ **to** n **do**

$$\bar{a}_j = \bar{a}_j - \left(\frac{A_{j,i}}{A_{i,i}}\right) \bar{a}_i$$

    end

end

Algorithm 2: Gaussian elimination with partial pivoting (part 2).

// determines if $\mathbf{A}$ is singular (if the bottom right element of $\mathbf{A}$ is 0 (to within machine precision), then the entire bottom row is 0 and $\mathbf{A}$ is singular)

if $|A_{n,n}| \leq \varepsilon$ **then**
  $\text{singular} = \text{true}$
end

Preallocate/initialize the solution vector $\mathbf{x} \in \mathbb{R}^{n \times 1}$.

// perform backward substitution to solve $\mathbf{Ax} = \mathbf{b}$ if $\mathbf{A}$ is nonsingular

if !(singular) **then**
  $x_n = \dfrac{A_{n,n+1}}{A_{n,n}}$
  for $i = n - 1$ **to** 1 **by** -1 **do**
    $S = 0$ **for** $i + 1$ **to** n **do**
      $S = S + A_{i,j} x_j$
    end
    $x_i = \dfrac{A_{i,n+1} - S}{A_{i,i}}$
  end
end

References

[1] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. 9^{th}. Boston, MA: Brooks/Cole, Cengage Learning, 2011.