Gaussian elimination

Consider solving the linear system with 3 eqns and 3 unknowns

$$\begin{aligned} 2x_1 + 4x_2 + 2x_3 &= 4 \quad (1)\\ 4x_1 + 7x_2 + 7x_3 &= 13 \quad (2)\\ -2x_1 - 7x_2 + 5x_3 &= 7 \quad (3) \end{aligned}$$

Row elimination: multiples of one row are subtracted from another row.
Eliminate x_1 from (2), (3).

$$\begin{aligned} 2x_1 + 4x_2 + 2x_3 &= 4\\ -x_2 + 3x_3 &= 5 \quad (4)\\ -3x_2 + 7x_3 &= 11 \quad (5) \end{aligned}$$

Here (2)-2×(1)→(4) and (3)-(-1)×(1)→(5). Eliminate x_2 from (5).

$$\begin{aligned} 2x_1 + 4x_2 + 2x_3 &= 4\\ -x_2 + 3x_3 &= 5\\ -2x_3 &= -4 \quad (6) \end{aligned}$$

Here (5)-3×(4)→(6).

(Back) substitution

$x_3 = -4/-2 = 2$
$-x_2 = 5 - 3 \cdot 2 = -1 \Rightarrow x_2 = 1$
$2x_1 = 4 - 2 \cdot 2 - 4 \cdot 1 = -4 \Rightarrow x_1 = -2$

There are two 3×3 matrices associated with elimination.
The upper $\triangle$ matrix U and the lower $\triangle$ matrix L.

$$U = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -1 & 3 \\ 0 & 0 & -2 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

The elimination factors $2, -1, 3$ are called **multipliers**. Note the 1's on the diagonal of L. If A represents the above linear system then A is related to matrices L and U with the relation $A = LU$.

Gaussian elimination (GE)

Eliminate unknowns x_1 through x_{n-1}.

```
for k = 1 to n-1
    eliminate x_k from rows k+1 to n
end for
```

or

```
for k = 1 to n-1
    for i = k+1 to n
        eliminate x_k from row i
    end for
end for
```


How to eliminate?

```
for k = 1 to n-1
    for i = k+1 to n
        multiply row k with appropriate factor
           and subtract result from row i
        set result to row i
    end for
end for
```

```
for k = 1 to n-1
    for i = k+1 to n
        t = a_ik/a_kk
        for j = 1 to n
            a_ij = a_ij - t · a_kj
        end for
    end for
end for
```

Why not save computation?

for $k = 1$ to $n-1$
    for $i = k+1$ to n
        $t = a_{ik}/a_{kk}$
        for $j = k+1$ to n
            $a_{ij} = a_{ij} - t \cdot a_{kj}$
        end for
    end for
end for

What if we want to store the multipliers in L?

for $k = 1$ to $n-1$
    for $i = k+1$ to n
        $l_{ik} = a_{ik}/a_{kk}$
        for $j = k+1$ to n
            $a_{ij} = a_{ij} - l_{ik} \cdot a_{kj}$
        end for
    end for
end for

Why not save space?
GE (without pivoting) algo for $n \times n$ matrices

for $k = 1$ to $n-1$
    for $i = k+1$ to n
        $a_{ik} = a_{ik}/a_{kk}$ *a_{ik} multiplier*
        for $j = k+1$ to n
            $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
        end for
    end for
end for

Above algo overwrites strictly lower triangular part of A by L and upper triangular part of A with U. The 1's on diagonal of L are not stored.

Operation counts:
$\sum_{k=1}^{n-1}(n-k)^2 = \sum_{k=1}^{n-1} k^2 \approx \frac{n^3}{3}$ additions and multiplications (flops) for innermost loop statement.

How about the rhs vector?
GE (without pivoting) algo for $n \times n$ matrices with simultaneous processing of the rhs vector

for $k = 1$ to $n-1$
    for $i = k+1$ to n
        $a_{ik} = a_{ik}/a_{kk}$
        $b_i = b_i - a_{ik}b_k$
        for $j = k+1$ to n
            $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
        end for
    end for
end for

The additional work for processing the rhs vector is $\sum_{k=1}^{n-1}(n-k) = \sum_{k=1}^{n-1} k \approx \frac{n^2}{2}$ flops.

Back substitution

Assume that we have applied GE to the matrix A with simultaneous processing of rhs vector b. Then, the soln of linear system $Ax = b$ is obtained by back substitution (BS), as follows:

for $i = n$ down to 1
    compute $x_i = (b_i - \sum_{j=i+1}^{n} a_{ij}x_j)/a_{ii}$
end for

BS algo for $n \times n$ upper-triangular matrices

for $i = n$ down to 1
    $x_i = b_i$
    for $j = i+1$ to n
        $x_i = x_i - a_{ij}x_j$
    end for
    $x_i = x_i/a_{ii}$
end for

Operation counts:
$\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1} i \approx \frac{n^2}{2}$ flops for innermost loop statement.

LU factorization

Assume we apply GE algo to A.
Let L and U be the lower and upper triangular matrices, respectively, obtained by GE.
It can be shown that $A = LU$. The factorization of A into L and U is referred to as **LU factorization**.

What if during GE the processing of rhs vector is not performed simultaneously with the processing of the matrix?
$A = LU, Ax = b \Rightarrow LUx = b \Rightarrow Ly = b, Ux = y$.

That is, solving $Ly = b$, then $Ux = y$. Since both of these systems are triangular we can first use forward substitution then backward substitution algo.
If we want to process more than one linear system with the same A and different rhs we would have to perform LU factorization once and do 2 substitutions for each rhs.

Forward substitution

Assume A is a lower triangular matrix. Then, soln of linear system $Ax = b$ is obtained by forward substitution (FS), as follows:

for $i = 1$ to n
    compute $x_i = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j)/a_{ii}$
end for

FS algo for $n \times n$ lower-triangular matrices

```
for i = 1 to n
    x_i = b_i
    for j = 1 to i − 1
        x_i = x_i − a_ij x_j
    end for
    x_i = x_i / a_ii
end for
```

Operation counts:
$\sum_{i=1}^{n-1} i \approx \frac{n^2}{2}$ flops for innermost loop statement. Note that the division statement is not required for unit lower triangular matrices.

Partial pivoting

Solve

$$\begin{aligned} .001x_1 + x_2 &= 1 \quad (7) \\ x_1 + x_2 &= 2 \quad (8) \end{aligned}$$

To eliminate x_1, we want (8)-1000(7).
But in 2 decimal digits (rounding) we have
$fl(1 - 1000) = fl(-999) = -.10 \times 10^4$
$fl(2 - 1000) = fl(-998) = -.10 \times 10^4$
Hence,

$$\begin{aligned} .001x_1 + x_2 &= 1 \\ x_2 &= 1 \end{aligned}$$

$\Rightarrow x_1 = 0$.
But with exact arithmetic we have

$$\begin{aligned} .001x_1 + x_2 &= 1 \\ 999x_2 &= 998 \end{aligned}$$

$\Rightarrow x_2 \approx .999, x_1 \approx 1.00$

One way to improve the accuracy of the soln is to interchange rows and work with

$$\begin{aligned} x_1 + x_2 &= 2 \quad (9) \\ .001x_1 + x_2 &= 1 \quad (10) \end{aligned}$$

Solve (10)-.001×(9).
In 2 decimal digits (rounding)

$$\begin{aligned} x_1 + x_2 &= 2 \\ x_2 &= 1 \end{aligned}$$

$\Rightarrow x_2 = 1, x_1 = 1$ which agrees well.

The inaccuracy in our first solution was due to the small coefficient of x_1 in row 1. Since $x_2 \approx 1$ and $x_1 = \frac{1-x_2}{.001} = 1000(1 - x_2)$ the cancellation that happens in $1 - x_2$ propagate by a 1000. To avoid small diagonal elements we follow the procedure called (row) partial pivoting.

GE with partial pivoting

for $k = 1$ to $n-1$
    $l = arg \max_{i=k}^{n}\{|a_{ik}|\}$ *pivot*
    if $a_{lk} = 0$, system is singular, quit
    interchange rows k and l
    for $i = k+1$ to n
        $a_{ik} = a_{ik}/a_{kk}$
        for $j = k+1$ to n
            $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
        end for
    end for
end for

This requires about the same amount of work as the no-pivoting algorithm ($\frac{n^3}{3}$).

Scaling

There are cases that partial pivoting is not enough to generate an accurate solution. Consider

$$\begin{aligned} 2x_1 + 2000x_2 &= 2000 \\ x_1 + x_2 &= 2 \end{aligned}$$

Partial pivoting will not change the order of the eqns, since 2 is the largest coef in col 1. But row elimination will give us the results of the no pivoting process. One way to improve the accuracy is to normalize each row by dividing them by the largest coef.

GE with scaled partial pivoting

for $i = 1$ to n
    $s = \max_{j=1}^{n}\{|a_{ij}|\}$
    if $s = 0$, system is singular, quit
    for $j = 1$ to n
        $a_{ij} = a_{ij}/s$
    end for
end for

Now apply GE with partial pivoting.

Complete pivoting

Before the k-th elimination step, interchange row k with row i and col k with col j such that a_{ij} is the absolutely largest coef in the partly processed matrix.

GE with complete pivoting

for $k = 1$ to $n-1$
    $l, m = arg \max_{i=k,j=k}^{n}\{|a_{ij}|\}$ *pivot*
    if $a_{lm} = 0$, system is singular, quit
    interchange rows k and l, then cols k and m
    for $i = k+1$ to n
        $a_{ik} = a_{ik}/a_{kk}$
        for $j = k+1$ to n
            $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
        end for
    end for
end for

This requires about the same amount of work as the no-pivoting algorithm ($\frac{n^3}{3}$). But usually, the extra work is not worth the accuracy improvement. Partial pivoting is more frequently used.

Symmetric matrices

Consider $a_{ij} = a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj}$ and $a_{ji} = a_{ji} - \frac{a_{jk}}{a_{kk}} a_{ki}$
Since $a_{kj} = a_{jk}$ and $a_{ki} = a_{ik}$, updates of a_{ij} and a_{ji} are same. So, every elimination step preserves symmetry of submatrix.

$l_{ik} = \frac{a_{ik}}{a_{kk}} = \frac{a_{ki}}{a_{kk}}$, $i > k$.
After step k, col k of multiplier matrix L is same as row k of matrix U divided by diagonal element.

GE (without pivoting) algo for symmetric matrices

for $k = 1$ to $n-1$
  for $i = k+1$ to n
    $t = a_{ki}/a_{kk}$
    for $j = i$ to n
      $a_{ij} = a_{ij} - t \cdot a_{kj}$
    end for
  end for
end for

This algo overwrites the upper triangular part of A with $U = DL^T$. $L = U^T D^{-1}$.

Asymptotically, it requires $\frac{n^3}{6}$ flops i.e. half of what GE/LU on general $n \times n$ systems requires.

Banded matrices

GE (without pivoting) algo for (l, u)-banded matrices

for $k = 1$ to $n-1$
  for $i = k+1$ to min$\{k+l, n\}$
    $a_{ik} = a_{ik}/a_{kk}$
    for $j = k+1$ to min$\{k+u, n\}$
      $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
    end for
  end for
end for

This algo overwrites strictly lower triangular part of A by L and upper triangular part of A with U. The 1's on diagonal of L are not stored.

At each step k, the algo works on a $(l+1)\times(u+1)$ submatrix. In each of the last $l-1$ or $u-1$ steps the size of the submatrix decreases by 1, so that the algorithm does not go out of bounds.

The algorithm requires$\approx \sum_{k=1}^{n-1}(l+1)(u+1) \approx \sum_{k=1}^{n-1} lu \approx nlu$ flops.

FS and BS for banded matrices

FS algo for (l,0)-banded matrices

```
for i = 1 to n
    x_i = b_i
    for j = max{i − l, 1} to i − 1
        x_i = x_i − a_ij x_j
    end for
    x_i = x_i / a_ii
end for
```

BS algo for (0,u)-banded matrices

```
for i = n down to 1
    x_i = b_i
    for j = i + 1 to min{i + u, n}
        x_i = x_i − a_ij x_j
    end for
    x_i = x_i / a_ii
end for
```

FS requires $\sum_{i=1}^{n}(l+1) \approx nl$ flops and
BS requires $\sum_{i=1}^{n}(u+1) \approx nu$ flops.
Therefore, the soln of an $n \times n(l,u)$-banded linear system by banded LU factorization and BS and FS requires $nlu + n(u+l) \approx nlu$ flops.

Sparse matrices and storage schemes

Let A be an $n \times m$ sparse matrix.
Yale, Gustavson:
A is represented by 3 1D arrays. Let nz be the total number of non-zero elements in A.
• Array a, of size nz, holds all non-zero elements of A row-by-row.
• Array ja, of size nz, gives col index in A for each element of a.
• Array ia, of size $n+1$, gives the location in a of the first non-zero element in each row of A.

Profile:
A is represented by 3 1D arrays.
• Array a holds the elements in each row of A starting from the first non-zero and ending at the last non-zero, including 0's in between.
• Array ja, of size n, gives the col index in A of the first non-zero element in each row of A.
• Array ia, of size $n+1$, gives the location in a of the first non-zero element in each row of A.

ELLPACK, Purdue, primary:
A is represented by 2 2D arrays. Let $mnzr$ be the maximum number of non-zero elements in any row of A. All the arrays used are of size $n \times mnzr$
• Array a has each row holding all non-zero elements of the respective row of A.
• Array ja gives the col index in A for each element in a.

MATLAB:
A is represented by 3 1D arrays. Let nz be the total number of non-zero elements in A.
• Array a, of size nz, holds all non-zero elements of A col-by-col.
• Array ja, of size nz, gives col index in A for each element of a.
• Array ia, of size nz, gives row index in A for each element of a.