Solution of linear systems

Solving linear systems lies in the heart of almost any scientific, engineering, financial, and not only, problem.

Let A be $n \times n$ and b be $n \times 1$. Solving $Ax = b$ means computing an $n \times 1$ vector x, that satisfies $Ax = b$.

We are interested in methods for computing x, given A and b, and more particular in methods that compute x with high accuracy (small error), and high efficiency (low number of operations).

The most fundamental method for solving linear systems is **Gauss elimination** (GE). This is based in the general technique of transforming a given linear system to another, mathematically equivalent to the first (same solution with the first), but easier to solve (using a self-explanatory method).

We will first see how the ‘‘easy’’ linear systems are solved, and then how we transform other systems to easy systems.

Solution of lower triangular systems

Let A be $n \times n$ lower triangular matrix, with all diagonal elements nonzero. Let b be $n \times 1$. Let’s solve $Ax = b$.

Write all equations arising from $Ax = b$:

$$\begin{array}{cccccccccc}
a_{11}x_1 & & & & & & & & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & & & & & & = & b_2 \\
. & . & . & & & & & & . & . \\
a_{i1}x_1 & + & a_{i2}x_2 & + \cdots + & a_{ii}x_i & & & & = & b_i \\
. & . & . & . & . & . & & & . & . \\
a_{n1}x_1 & + & a_{n2}x_2 & + \cdots + & a_{ni}x_i & + \cdots + & a_{nn}x_n & & = & b_n
\end{array}$$

Starting with the first equation, we compute the first unknown, substitute in the second equation, compute the second unknown, substitute in the third equation, etc., proceeding forward to the last equation. Thus we have

Solution of lower triangular systems

$$\begin{array}{ccc}
x_1 & = & b_1/a_{11} \\
x_2 & = & (b_2 - a_{21}x_1)/a_{22} \\
. & . & . \\
x_i & = & (b_i - \sum_{k=1}^{i-1} a_{ik}x_k)/a_{ii} \\
. & . & . \\
x_n & = & (b_n - \sum_{k=1}^{n-1} a_{nk}x_k)/a_{nn}
\end{array}$$

The algorithm that describes the above procedure is

```
for $i$ = 1 to $n$ do
    if $a_{ii} \neq 0$, $x_i = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j)/a_{ii}$, else quit
endfor
```

or

Forward substitution algorithm for $n \times n$ lower-triangular matrices

Forward substitution algorithm for $n \times n$ lower-triangular matrices

```
for $i$ = 1 to $n$ do
    $x_i = b_i$
    for $j$ = 1 to $i$-1 do
        $x_i = x_i - a_{ij}x_j$
    endfor
    if $a_{ii} \neq 0$, $x_i = x_i \,/\, a_{ii}$, else quit
endfor
```

How many operations are needed? About $1 + 2 + \cdots + (n-1) = n(n-1)/2 \approx n^2/2$ pairs of additions and multiplications (flops) and n divisions. Since divisions are much less than additions and multiplications, we often ignore them in the operation counts.

Solution of upper triangular systems

Let A be $n \times n$ upper triangular matrix, with all diagonal elements nonzero. Let b be $n \times 1$. Let's solve $Ax = b$.

Write all equations arising from $Ax = b$:

$$\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & +\cdots+ & a_{1j}x_j & +\cdots+ & a_{1n}x_n & = & b_1 \\
 & & a_{22}x_2 & +\cdots+ & a_{2j}x_j & +\cdots+ & a_{2n}x_n & = & b_2 \\
 & & & & . & . & . & . & . \\
 & & & & a_{jj}x_j & +\cdots+ & a_{jn}x_n & = & b_j \\
 & & & & & & . & . & . \\
 & & & & & & a_{nn}x_n & = & b_n
\end{array}$$

Starting with the last (nth) equation, we compute the last unknown, substitute in the equation before the last ($n-1$st), compute the unknown before the last, substitute in the equation before the second last, etc., proceeding backward to the first equation.

Backward substitution algorithm for $n \times n$ upper-triangular matrices

The algorithm that describes the above procedure is

```
for $i = n$ down to 1 do
    if $a_{ii} \neq 0$, $x_i = (b_i - \sum_{j=i+1}^{n} a_{ij}x_j)/a_{ii}$, else quit
endfor
```

or

Back substitution algorithm for $n \times n$ upper-triangular matrices

```
for $i = n$ down to 1 do
    $x_i = b_i$
    for $j = i+1$ to $n$ do
        $x_i = x_i - a_{ij}x_j$
    endfor
    if $a_{ii} \neq 0$, $x_i = x_i / a_{ii}$, else quit
endfor
```

How many operations are needed? About $n^2/2$ pairs of additions and multiplications (flops) and n divisions.

Equivalent linear systems -- row operations

Two linear systems are called **equivalent**, when they admit exactly the same solutions (or sets of solutions). That is, $Ax = b \iff By = c$ when $x = y$.
Given a linear system, there are infinitely many equivalent to it.
One way to obtain systems equivalent to a given one is to apply the so-called **row operations** to the initial given system. There are three main row operations:
(1) scalar multiplication of a row (equation) ρ_i with nonzero scalar $\rho_i \leftarrow \kappa\rho_i$, $\kappa \neq 0$
(2) addition of rows (equations) $\rho_i \leftarrow \kappa\rho_i + \lambda\rho_j$, $\kappa \neq 0$, $\lambda \neq 0$
(3) permutation of rows (equations) $\rho_i \longleftrightarrow \rho_j$

Gauss elimination (the primary method for solving linear systems) is based on these row operations. With a series of row operations, the initial system is transformed into an equivalent upper triangular one.

Gauss elimination method -- example

Consider the linear system

$$\begin{array}{ccrrrrcr}
\rho_1 & := & x_1 & -2x_2 & +x_3 & +x_4 & = & 5 \\
\rho_2 & := & 2x_1 & -x_2 & +5x_3 & -4x_4 & = & -5 \\
\rho_3 & := & -x_1 & +3x_2 & -x_3 & +x_4 & = & 1 \\
\rho_4 & := & -3x_1 & +7x_2 & -5x_3 & +x_4 & = & -10
\end{array}$$

or $Ax = b$, where

$$A = \begin{bmatrix} 1 & -2 & 1 & 1 \\ 2 & -1 & 5 & -4 \\ -1 & 3 & -1 & 1 \\ -3 & 7 & -5 & 1 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ -5 \\ 1 \\ -10 \end{bmatrix}.$$

The system can be also described with the so-called **augmented** matrix

$$[A:b] = \begin{bmatrix} 1 & -2 & 1 & 1 & : & 5 \\ 2 & -1 & 5 & -4 & : & -5 \\ -1 & 3 & -1 & 1 & : & 1 \\ -3 & 7 & -5 & 1 & : & -10 \end{bmatrix}.$$

Consider the row operations $\rho_2^{(1)} \leftarrow \rho_2 - 2\rho_1$, $\rho_3^{(1)} \leftarrow \rho_3 + \rho_1$, $\rho_4^{(1)} \leftarrow \rho_4 + 3\rho_1$

Gauss elimination method -- example -- step 1

Elimination of x_1 from rows (equations) 2 to 4, with the row operations
$\rho_2^{(1)} \leftarrow \rho_2 - 2\rho_1\,, \quad \rho_3^{(1)} \leftarrow \rho_3 - (-\rho_1)\,, \quad \rho_4^{(1)} \leftarrow \rho_4 - (-3\rho_1)$
The initial system is transformed to the equivalent one

$$\begin{array}{lcrrrrcr}
\rho_1^{(1)} & := & x_1 & -2x_2 & +x_3 & +x_4 & = & 5 \\
\rho_2^{(1)} & := & & 3x_2 & +3x_3 & -6x_4 & = & -15 \\
\rho_3^{(1)} & := & & x_2 & 0x_3 & +2x_4 & = & 6 \\
\rho_4^{(1)} & := & & x_2 & -2x_3 & +4x_4 & = & 5
\end{array}$$

or

$$[A^{(1)} \colon b^{(1)}] = \begin{bmatrix} 1 & -2 & 1 & 1 & : & 5 \\ 0 & 3 & 3 & -6 & : & -15 \\ 0 & 1 & 0 & 2 & : & 6 \\ 0 & 1 & -2 & 4 & : & 5 \end{bmatrix}.$$

How were the factors $2, -1, -3$ chosen in the above row operations? Note that
$2 = \dfrac{2}{1} = \dfrac{a_{21}}{a_{11}}, \quad -1 = \dfrac{-1}{1} = \dfrac{a_{31}}{a_{11}}, \quad -3 = \dfrac{-3}{1} = \dfrac{a_{41}}{a_{11}}$, so that the coefficients in positions a_{21}, a_{31}, a_{41} are annihilated. Element a_{11} (which we divide with) is called **pivot** for step 1. Next consider $\rho_3^{(2)} \leftarrow \rho_3^{(1)} - \dfrac{1}{3}\rho_2^{(1)}\,, \quad \rho_4^{(2)} \leftarrow \rho_4^{(1)} - \dfrac{1}{3}\rho_2^{(1)}$

Gauss elimination method -- example -- step 2

Elimination of x_2 from rows (equations) 3 to 4, with the row operations
$\rho_3^{(2)} \leftarrow \rho_3^{(1)} - \dfrac{1}{3}\rho_2^{(1)}\,, \quad \rho_4^{(2)} \leftarrow \rho_4^{(1)} - \dfrac{1}{3}\rho_2^{(1)}$
The system $[A^{(1)} \colon b^{(1)}]$ is transformed to the equivalent one

$$\begin{array}{lcrrrrcr}
\rho_1^{(2)} & := & x_1 & -2x_2 & +x_3 & +x_4 & = & 5 \\
\rho_2^{(2)} & := & & 3x_2 & +3x_3 & -6x_4 & = & -15 \\
\rho_3^{(2)} & := & & & -x_3 & +4x_4 & = & 11 \\
\rho_4^{(2)} & := & & & -3x_3 & +6x_4 & = & 10
\end{array}$$

or

$$[A^{(2)} \colon b^{(2)}] = \begin{bmatrix} 1 & -2 & 1 & 1 & : & 5 \\ 0 & 3 & 3 & -6 & : & -15 \\ 0 & 0 & -1 & 4 & : & 11 \\ 0 & 0 & -3 & 6 & : & 10 \end{bmatrix}.$$

How were the factors $\dfrac{1}{3}, \dfrac{1}{3}$ chosen in the above row operations? Note that
$\dfrac{1}{3} = \dfrac{a_{32}^{(1)}}{a_{22}^{(1)}}, \quad \dfrac{1}{3} = \dfrac{a_{42}^{(1)}}{a_{22}^{(1)}}$, so that the coefficients in positions $a_{32}^{(1)}, a_{42}^{(1)}$ are annihilated. $a_{32}^{(1)}, a_{42}^{(1)}$. Element $a_{22}^{(1)}$ is called **pivot** for step 2. Next do $\rho_4^{(3)} \leftarrow \rho_4^{(2)} - 3\rho_3^{(2)}$

Gauss elimination method -- example -- step 3

Elimination of x_3 from row (equation) 4, with the row operation
$\rho_4^{(3)} \leftarrow \rho_4^{(2)} - 3\rho_3^{(2)}$
The system $[A^{(2)} \colon b^{(2)}]$ is transformed to the equivalent one

$$\begin{array}{lcrrrrcr}
\rho_1^{(3)} & := & x_1 & -2x_2 & +x_3 & +x_4 & = & 5 \\
\rho_2^{(3)} & := & & 3x_2 & +3x_3 & -6x_4 & = & -15 \\
\rho_3^{(3)} & := & & & -x_3 & +4x_4 & = & 11 \\
\rho_4^{(3)} & := & & & & -6x_4 & = & -23
\end{array}$$

or

$$[A^{(3)} \colon b^{(3)}] = \begin{bmatrix} 1 & -2 & 1 & 1 & : & 5 \\ 0 & 3 & 3 & -6 & : & -15 \\ 0 & 0 & -1 & 4 & : & 11 \\ 0 & 0 & 0 & -6 & : & -23 \end{bmatrix}.$$

How was the factor 3 chosen in the above row operation? Note that
$3 = \dfrac{-3}{-1} = \dfrac{a_{43}^{(2)}}{a_{33}^{(2)}}$, so that the coefficient in position $a_{43}^{(2)}$ is annihilated. Element $a_{33}^{(2)}$ is called **pivot** for step 3.

Gauss elimination method -- designing the algorithm

To describe the algorithm for Gauss elimination, we describe each step, using a top-down approach, first with fewer, then with more details, so that we eventually get a high-level pseudo-code.

General description: During the kth step of Gauss elimination, the unknown x_k is eliminated from rows $k+1$ to n. This is done for $k = 1, \cdots, n-1$.

More specifically,

```
for $k$ = 1 to $n - 1$ do
    eliminate $x_k$ from rows $k$+1 ... $n$
endfor
```

or

```
for $k$ = 1 to $n - 1$ do
    for $i$ = $k$+1 to $n$ do
        eliminate $x_k$ from row $i$
    endfor
endfor
```

Gauss elimination method -- designing the algorithm

How is elimination done?

```
for $k = 1$ to $n-1$ do
    for $i = k+1$ to $n$ do /* $\rho_i^{(0)} \leftarrow \rho_i$ */
        $\rho_i^{(k)} \leftarrow \rho_i^{(k-1)} - \lambda \rho_k^{(k-1)}$ /* appropriate $\lambda$ multiplier
    endfor
endfor
```

or in more details (specify multiplier)

```
$A^{(0)} = A$
for $k = 1$ to $n-1$ do
    for $i = k+1$ to $n$ do
        if $a_{kk}^{(k-1)} \neq 0$, $t = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}$ else quit
        for $j = 1$ to $n$ do
            $a_{ij}^{(k)} = a_{ij}^{(k-1)} - t \cdot a_{kj}^{(k-1)}$
        endfor
    endfor
endfor
```

Gauss elimination method -- designing the algorithm

Saving memory (no need to store all intermediate instances of A)

```
for $k = 1$ to $n-1$ do
    for $i = k+1$ to $n$ do
        if $a_{kk} \neq 0$, $t = a_{ik} / a_{kk}$, else quit
        for $j = 1$ to $n$ do
            $a_{ij} = a_{ij} - t \cdot a_{kj}$
        endfor
    endfor
endfor
```

Saving time (operations) (no need to compute zeros)

```
for $k = 1$ to $n-1$ do
    for $i = k+1$ to $n$ do
        if $a_{kk} \neq 0$, $t = a_{ik} / a_{kk}$, else quit
        for $j = k+1$ to $n$ do
            $a_{ij} = a_{ij} - t \cdot a_{kj}$
        endfor
    endfor
endfor
```

Gauss elimination method -- designing the algorithm

As we will see later, it is useful to store the multipliers. These can be stored in a strictly lower triangular matrix.

Storing the multipliers in a strictly lower triangular matrix L:

```
for $k = 1$ to $n-1$ do
    for $i = k+1$ to $n$ do
        if $a_{kk} \neq 0$, $l_{ik} = a_{ik} / a_{kk}$, else quit
        for $j = k+1$ to $n$ do
            $a_{ij} = a_{ij} - l_{ik} \cdot a_{kj}$
        endfor
    endfor
endfor
```

The above scheme uses a whole matrix (L) for storing the multipliers, while only the strictly lower triangular part of the matrix is needed.

Furthermore, the strictly lower triangular part of A is not going to be used again anymore. Thus, to save space (memory), we can store the multipliers, i.e. the strictly lower triangular part of L, in the respective part of A.

Gauss elimination (without pivoting) algorithm for general $n \times n$ matrices

Saving memory (again) ... final form

Gauss elimination (without pivoting) algorithm for general $n \times n$ matrices

```
for $k = 1$ to $n$-1 do
    for $i = k+1$ to $n$ do
        if $a_{kk} \neq 0$, $a_{ik} = a_{ik} / a_{kk}$, else quit    /* $a_{kk}$ pivot */
        for $j = k+1$ to $n$ do
            $a_{ij} = a_{ij} - a_{ik} a_{kj}$   /* $a_{ik}$ multiplier */
        endfor
    endfor
endfor
```

The above algorithm overwrites the strictly lower triangular part of A by the strictly lower triangular part of L (the multipliers) and the upper triangular part of A with new numbers, that can be considered to form an upper triangular matrix U.

How many operations are required? About $\sum_{k=1}^{n-1} (n-k)^2 = \sum_{k=1}^{n-1} k^2$ $= \dfrac{n(n-1)(2n-1)}{6} \approx \dfrac{n^3}{3}$ pairs of additions and mult. (flops), and $\dfrac{n^2}{2}$ divisions.

Gauss elimination (without pivoting) algorithm for general $n \times n$ matrices (cont.)

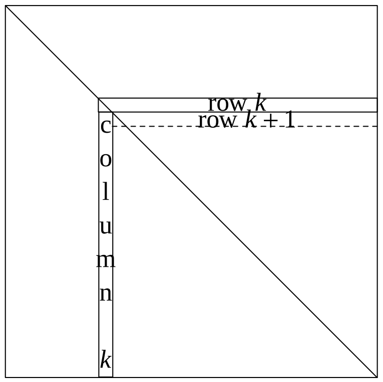
Since divisions are much less than additions and multiplications, we often ignore them in the operation counts.

There exist variations of this algorithm with different ordering of the i, j and k loops, giving rise to the kji, ijk, etc, versions.

The above algorithm is applied to general $n \times n$ matrices, that is, square matrices without particular properties. If the matrix has particular properties, such as symmetry, bandedness, etc, then the algorithm can be adjusted to take advantage of the special properties, so that it requires fewer operations.

The kth step of Gauss elimination w/out pivoting alg. for general $n \times n$ matrices

During the kth step of the Gauss elimination algorithm changes are applied to the lower-right submatrix of size $(n-k) \times (n-k)$, as well as to the kth column, rows $k+1$ to n.



- Row k (columns k to n) is read (but not written).
- Column k (rows $k+1$ to n) is updated for the last time and stores multipliers.
  $a_{ik} = a_{ik} / a_{kk}$ (a_{kk} pivot a_{ik} mult.)
- The submatrix is updated. An appropriate multiple of row k is subtracted from each other row of the submatrix.
  $a_{ij} = a_{ij} - a_{ik} a_{kj}$
  - Row $k+1$ of the submatrix (columns $k+1$ to n) is updated for the last time.
    $a_{k+1,j} = a_{k+1,j} - a_{k+1,k} a_{k,j}$
  - The rest of the submatrix rows are updated. They will be updated again in the next step.

Simultaneous processing of the right-hand side vector

Gauss elimination (without pivoting) algorithm for general $n \times n$ matrices with simultaneous processing of the right-hand side vector

```
for $k$ = 1 to $n$-1 do
    for $i$ = $k$+1 to $n$ do
        if $a_{kk} \neq 0$, $a_{ik} = a_{ik} / a_{kk}$, else quit     /* $a_{kk}$ pivot */
        for $j$ = $k$+1 to $n$ do
            $a_{ij} = a_{ij} - a_{ik} a_{kj}$   /* $a_{ik}$ multiplier */
        endfor
        $b_i = b_i - a_{ik} b_k$
    endfor
endfor
```

The additional work for processing the right-hand side vector requires and extra $\sum_{k=1}^{n-1} (n-k) = \sum_{k=1}^{n-1} k = \frac{(n-1)n}{2} \approx \frac{n^2}{2}$ pairs of additions and multiplications (flops).

Imptt

Cost of solving a general linear system by Gauss elimination

Gauss elimination: $\frac{n^3}{3}$ pairs of additions and multiplications (flops) and $\frac{n^2}{2}$ divisions.

Simultaneous processing of the right-hand side vector: $\frac{n^2}{2}$ pairs of additions and multiplications (flops).

Back substitution: $\frac{n^2}{2}$ pairs of additions and multiplications (flops) and n divisions.

Total: $\frac{n^3}{3} + 2 \times \frac{n^2}{2}$ pairs of additions and multiplications (flops) and $\frac{n^2}{2} + n$ divisions.