Linear Systems

Example: Find x_1, x_2, x_3 such that the following three equations hold:

$$\begin{aligned} 2x_1 + 3x_2 + x_3 &= 1 \\ 4x_1 + 3x_2 + x_3 &= -2 \\ -2x_1 + 2x_2 + x_3 &= 6 \end{aligned}$$

We can write this using matrix-vector notation as

$$\underbrace{\begin{bmatrix} 2 & 3 & 1 \\ 4 & 3 & 1 \\ -2 & 2 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 1 \\ -2 \\ 6 \end{bmatrix}}_{b}$$

General case: We can have n equations for n unknowns:

Given: Coefficients $a_{11}, \dots, a_{nn}$, right hand side entries $b_1, \dots, b_n$.

Wanted: Numbers $x_1, \dots, x_n$ such that

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

Using matrix-vector notation this can be written as follows: Given a matrix $A \in \mathbb{R}^{n\times n}$ and a right-hand side vector $b \in \mathbb{R}^n$, find a vector $x \in \mathbb{R}^n$ such that

$$\underbrace{\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}}_{b}$$

Singular and Nonsingular Matrices

Definition: We call a matrix $A \in \mathbb{R}^{n\times n}$ *singular* if there exists a nonzero vector $x \in \mathbb{R}^n$ such that $Ax = 0$.

Example: The matrix $A = \begin{bmatrix} 1 & -2 \\ -2 & 4 \end{bmatrix}$ is singular since for $x = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ we have $Ax = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

The matrix $A = \begin{bmatrix} 1 & -2 \\ 0 & 4 \end{bmatrix}$ is nonsingular: $Ax = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ implies $x_2 = b_2/4$, $x_1 = b_1 + 2x_2$. Therefore $Ax = 0$ implies $x = 0$.

Observation: If A is singular, the linear system $Ax = b$ has either no solution or infinitely many solutions: As A is singular there exists a nonzero vector y with $Ay = 0$. If $Ax = b$ has a solution x , then $x + \alpha y$ is also a solution for any $\alpha \in \mathbb{R}$.

We will later prove: If A is nonsingular, then the linear system $Ax = b$ has a unique solution x for any given $b \in \mathbb{R}^n$.

We only want to consider problems where there is a unique solution, i.e. where the matrix A is nonsingular. How can we check whether a given matrix A is nonsingular? If we use exact arithmetic we can use Gaussian elimination with pivoting (which will be explained later) to show that A is nonsingular. But in machine arithmetic we can only assume that a machine approximation for the matrix A is known, and we will have to use a different method to decide whether A is nonsingular in this case.

Gaussian Elimination without Pivoting

Basic Algorithm: We can add (or subtract) a multiple of one equation to another equation, without changing the solution of the linear system. By repeatedly using this idea we can eliminate unknowns from the equations until we finally get an equation which just contains one unknown variable.

1. **Elimination:**

   step 1: eliminate x_1 from equation 2 , … , equation n by subtracting multiples of equation 1:
   $\text{eq}_2 := \text{eq}_2 - \ell_{21} \cdot \text{eq}_1$, … , $\text{eq}_n := \text{eq}_n - \ell_{n1} \cdot \text{eq}_1$

   step 2: eliminate x_2 from equation 3 , … , equation n by subtracting multiples of equation 2:
   $\text{eq}_3 := \text{eq}_3 - \ell_{32} \cdot \text{eq}_2$, … , $\text{eq}_n := \text{eq}_n - \ell_{n2} \cdot \text{eq}_2$

   ⋮

   step $n-1$: eliminate x_{n-1} from equation n by subtracting a multiple of equation $n-1$:
   $\text{eq}_n := \text{eq}_n - \ell_{n,n-1} \cdot \text{eq}_{n-1}$

2. **Back substitution:**
   Solve equation n for x_n .
   Solve equation $n-1$ for x_{n-1} .
   ⋮
   Solve equation 1 for x_1 .

The elimination transforms the original linear system $Ax = b$ into a new linear system $Ux = y$ with an upper triangular matrix U , and a new right hand side vector y .

Example: We consider the linear system

$$\underbrace{\begin{bmatrix} 2 & 3 & 1 \\ 4 & 3 & 1 \\ -2 & 2 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 1 \\ -2 \\ 6 \end{bmatrix}}_{b}$$

Elimination: To eliminate x_1 from equation 2 we choose $l_{21} = 4/2 = 2$ and subtract l_{21} times equation 1 from equation 2. To eliminate x_1 from equation 3 we choose $l_{31} = -2/1 = -1$ and subtract l_{31} times equation 1 from equation 3. Then the linear system becomes

$$\begin{bmatrix} \textcircled{2} & 3 & 1 \\ 0 & -3 & -1 \\ 0 & 5 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \\ 7 \end{bmatrix}$$

To eliminate x_2 from equation 3 we choose $l_{32} = 5/(-3)$ and subtract l_{32} times equation 2 from equation 3, yielding the linear system

$$\underbrace{\begin{bmatrix} \textcircled{2} & 3 & 1 \\ 0 & \boxed{-3} & -1 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}}_{U} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \\ -4 \\ \frac{1}{3} \end{bmatrix}}_{y}$$

Now we have obtained a linear system with an upper triangular matrix (denoted by U) and a new right hand side vector (denoted by y).

Back substitution: The third equation is $\frac{1}{3}x_3 = \frac{1}{3}$ and gives $x_3 = 1$. Then the second equation becomes $-3x_2 - 1 = -4$, yielding $x_2 = 1$. Then the first equation becomes $2x_1 + 3 + 1 = 1$, yielding $x_1 = -\frac{3}{2}$.

Transforming the matrix and right hand side vector separately: We can split the elimination into two parts: The first part acts only on the matrix A , generating the upper triangular matrix U and the multipliers ℓ_{jk} . The second part uses the multipliers ℓ_{jk} to transform the right hand side vector b to the vector y .

1. **Elimination for matrix:** Given the matrix A , find an upper triangular matrix U and multipliers ℓ_{jk} :
   Let $U := A$ and perform the following operations on the rows of U :

$$\begin{array}{rl}
\text{step 1:} & \ell_{21} := u_{21}/u_{11},\ \text{row}_2 := \text{row}_2 - \ell_{21}\cdot\text{row}_1\ ,\ \dots\ ,\ \ell_{n1} := u_{n1}/u_{11},\ \text{row}_n := \text{row}_n - \ell_{n1}\cdot\text{row}_1 \\
\text{step 2:} & \ell_{32} := u_{32}/u_{22},\ \text{row}_3 := \text{row}_3 - \ell_{32}\cdot\text{row}_2\ ,\ \dots\ ,\ \ell_{n2} := u_{n2}/u_{22},\ \text{row}_n := \text{row}_n - \ell_{n2}\cdot\text{row}_2 \\
\vdots & \\
\text{step } n-1: & \ell_{n,n-1} := u_{n,n-1}/u_{n,n}, \quad \text{row}_n := \text{row}_n - \ell_{n,n-1}\cdot\text{row}_{n-1}
\end{array}$$

2. **Transform right hand side vector:** Given the vector b and the multiplies ℓ_{jk} find the transformed vector y :
   Let $y := b$ and perform the following operations on y :

$$\begin{array}{rl}
\text{step 1:} & y_2 := y_2 - \ell_{21}\cdot y_1\ ,\ \dots\ ,\ y_n := y_n - \ell_{n1}\cdot y_1 \\
\text{step 2:} & y_3 := y_3 - \ell_{32}\cdot y_2\ ,\ \dots\ ,\ y_n := y_n - \ell_{n2}\cdot y_2 \\
\vdots & \\
\text{step } n-1: & y_n := y_n - \ell_{n,n-1}\cdot y_{n-1}
\end{array}$$

3. **Back substitution:** Given the matrix U and the vector y find the vector x by solving $Ux = y$:
   $x_n := b_n/u_{n,n}$
   $x_{n-1} := (b_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1}$
   $\vdots$
   $x_1 := (b_1 - u_{12}x_2 - \cdots - u_{1n}x_n)/u_{11}$

What can possibly go wrong: Note that we have to divide by the so-called pivot elements $u_{11}, \dots, u_{n-1,n-1}$ in part 1, and we divide by $u_{11}, \dots, u_{nn}$ in part 3. If we encounter a zero pivot we have to stop the algorithm with an error message. Part 1 of the algorithm therefore becomes

$$\begin{array}{rl}
\text{step 1:} & \textbf{If } u_{11} = 0 \textbf{ then } \text{stop with error message } \textbf{else } \ell_{21} := u_{21}/u_{11},\ \text{row}_2 := \text{row}_2 - \ell_{21}\cdot\text{row}_1\ ,\ \dots \\
\text{step 2:} & \textbf{If } u_{22} = 0 \textbf{ then } \text{stop with error message } \textbf{else } \ell_{32} := u_{32}/u_{22},\ \text{row}_3 := \text{row}_3 - \ell_{32}\cdot\text{row}_2\ ,\ \dots \\
\vdots & \\
\text{step } n-1: & \textbf{If } u_{n-1,n-1} = 0 \textbf{ then } \text{stop with error message } \textbf{else } \ell_{n,n-1} := u_{n,n-1}/u_{n,n}, \text{row}_n := \text{row}_n - \ell_{n,n-1}\cdot\text{row}_{n-1} \\
 & \textbf{If } u_{n,n} = 0 \textbf{ then } \text{stop with error message}
\end{array}$$

Observation: If for a given matrix A part 1 of the algorithm finishes without an error message, then parts 2 and 3 work, and the linear system has a unique solution. Hence the matrix A must be nonsingular. However, the converse is not true: For the matrix $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ the algorithm stops immediately since $u_{11} = 0$, but the matrix A is nonsingular.

Reformulating part 2 as forward substitution: Now we want to express the relation between b and y in a different way. Assume we are given y , and we want to reconstruct b by reversing the operations: For $n = 3$ we would perform the operations

$$y_3 := y_3 + \ell_{32}y_2, \qquad y_3 := y_3 + \ell_{31}y_1, \qquad y_2 := y_2 + \ell_{21}y_1$$

and obtain

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 + \ell_{21}\cdot y_1 \\ y_3 + \ell_{31}\cdot y_1 + \ell_{32}\cdot y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix}}_{L} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

with the lower triangular matrix L .

Therefore we can rephrase the transformation step as follows: Given L and b , solve the linear system $Ly = b$ for y . Since L is lower triangular, we can solve this linear system by **forward substitution**: We first solve the first equation for y_1 , then solve the second equation for y_2 , … .

The LU decomposition: In the same way as we reconstructed the vector b from the vector y , we can reconstruct the matrix A from the matrix U : For $n=3$ we obtain

$$\begin{bmatrix} \text{row 1 of } A \\ \text{row 2 of } A \\ \text{row 3 of } A \end{bmatrix} = \begin{bmatrix} (\text{row 1 of } U) \\ (\text{row 2 of } U) + \ell_{2,1}\cdot(\text{row 1 of } U) \\ (\text{row 3 of } U) + \ell_{3,1}\cdot(\text{row 1 of } U) + \ell_{3,2}\cdot(\text{row 2 of } U) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix}}_{L} \begin{bmatrix} \text{row 1 of } U \\ \text{row 2 of } U \\ \text{row 3 of } U \end{bmatrix}$$

Therefore we have $A = LU$. We have written the matrix A as the product of a lower triangular matrix (with 1's on the diagonal) and an upper triangular matrix U . This is called the **LU-decomposition** of A .

Summary: Now we can rephrase the parts 1., 2., 3. of the algorithm as follows:

1. Find the LU-decomposition $A = LU$:
   Perform elimination on the matrix A , yielding an upper triangular matrix U . Store the multipliers in a matrix L and put 1's on the diagonal of the matrix L :

$$L := \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \ell_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n,1} & \cdots & \ell_{n,n-1} & 1 \end{bmatrix}$$

2. Solve $Ly = b$ using forward substitution
3. Solve $Ux = y$ using back substitution

The matrix decomposition $A = LU$ allows us to solve the linear system $Ax = b$ in two steps: Since

$$L\underbrace{Ux}_{y} = b$$

we can first solve $Ly = b$ to find y , and then solve $Ux = y$ to find x .

Example:

1. We start with $U := A$ and L being the zero matrix:

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

   step 1:

$$L = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} \textcircled{2} & 3 & 1 \\ 0 & \boxed{-3} & -1 \\ 0 & 5 & 2 \end{bmatrix}$$

   step 2:

$$L = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ -1 & -\frac{5}{3} & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} \textcircled{2} & 3 & 1 \\ 0 & \boxed{-3} & -1 \\ 0 & 0 & \boxed{\frac{1}{3}} \end{bmatrix}$$

   We put 1's on the diagonal of L and obtain $L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -\frac{5}{3} & 1 \end{bmatrix}, U = \begin{bmatrix} \textcircled{2} & 3 & 1 \\ 0 & \boxed{-3} & -1 \\ 0 & 0 & \boxed{\frac{1}{3}} \end{bmatrix}$.

2. We solve the linear system $Ly = b$
$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -\frac{5}{3} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 6 \end{bmatrix}$$
by forward substitution: The first equation gives $y_1 = 1$. Then the second equation becomes $2 + y_2 = -2$ yielding $y_2 = -4$. Then the third equation becomes $-1 - \frac{5}{3} \cdot (-4) + y_3 = 6$, yielding $y_3 = \frac{1}{3}$.

3. The back substitution for solving $Ux = b$ is performed as explained above.

Gaussian Elimination with Pivoting

There is a problem with Gaussian elimination without pivoting: If we have at step j that $u_{jj} = 0$ we cannot continue since we have to divide by u_{jj} . This element u_{jj} by which we have to divide is called the **pivot**.

Example: For $A = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 1 & 3 \\ 2 & -2 & 2 \end{bmatrix}$ we use $\ell_{21} = \frac{-2}{4}, \ell_{31} = \frac{2}{4}$ and obtain after step 1 of the elimination $U = \begin{bmatrix} 4 & -2 & 2 \\ 0 & 0 & 4 \\ 0 & -1 & 2 \end{bmatrix}$
Now we have $u_{22} = 0$ and cannot continue.

Gaussian elimination with pivoting uses row interchanges to overcome this problem. For step j of the algorithm we consider the **pivot candidates** $u_{j,j}, u_{j+1,j}, \ldots, u_{nj}$, i.e., the diagonal element and all elements below. If there is a nonzero pivot candidate, say u_{kj} we interchange rows j and k of the matrix U . Then we can continue with the elimination.

Since we want that the multipliers correspond to the appropriate row of U , we also **interchange the rows of** L whenever we interchange the rows of U . In order to keep track of the interchanges we use a vector p which is initially $(1, 2, \ldots, n)^\top$, and we **interchange the rows of** p whenever we interchange the rows of U .

Algorithm: Gaussian Elimination with pivoting: *Input:* matrix A . *Output:* matrix L (lower triangular), matrix U (upper triangular), vector p (contains permutation of $1, \ldots, n$)

$$L := \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} ; U := A ; p := \begin{bmatrix} 1 \\ \vdots \\ n \end{bmatrix}$$

```
For $j = 1$ to $n - 1$
    If $(U_{jj}, \ldots, U_{nj}) = (0, \ldots, 0)$
        Stop with error message “Matrix is singular”
    Else
        Pick $k \in \{j, j+1, \ldots, n\}$ such that $U_{kj} \neq 0$
    End
    Interchange row $j$ and row $k$ of $U$
    Interchange row $j$ and row $k$ of $L$
    Interchange $p_j$ and $p_k$
    For $k = j + 1$ to $n$
        $L_{kj} := U_{kj}/U_{jj}$
        $(\text{row } k \text{ of } U) := (\text{row } k \text{ of } U) - L_{kj} \cdot (\text{row } j \text{ of } U)$
    End
End
If $U_{nn} = 0$
    Stop with error message “Matrix is singular”
End
For $j = 1$ to $n$
    $L_{jj} := 1$
End
```

Note that we can have several nonzero pivot candidates, and we still have to specify which one we pick. This is called a **pivoting strategy**. Here are two examples:

Pivoting strategy 1: Pick the smallest k such that $U_{kj} \neq 0$.

Pivoting strategy 2: Pick k so that $|U_{kj}|$ is maximal.

If we use exact arithmetic, it does not matter which nonzero pivot we choose. However, in machine arithmetic the choice of the pivot can make a big difference for the accuracy of the result, as we will see below.

Example:

$$L = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} \textcircled{4} & -2 & 2 \\ -2 & 1 & 3 \\ 2 & -2 & 2 \end{bmatrix}, \qquad p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Here the pivot candidates are $4, -2, 2$, and we use 4 :

$$L = \begin{bmatrix} 0 & 0 & 0 \\ -\frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & 0 & 4 \\ 0 & \boxed{-1} & 1 \end{bmatrix}, \qquad p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Here the pivot candidates are $0, -1$, and we use -1 . Therefore we interchange rows 2 and 3 of L, U, p :

$$L = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & \boxed{-1} & 1 \\ 0 & 0 & 4 \end{bmatrix}, \qquad p = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

For column 2 we have $l_{32} = 0$ and U does not change. Finally we put 1 s on the diagonal of L and get the final result

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 4 & -2 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & 4 \end{bmatrix}, \qquad p = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

LU Decomposition for Gaussian elimination with pivoting: If we perform row interchanges during the algorithm we no longer have $LU = A$. Instead we obtain a matrix $\tilde{A}$ which contains the rows of A in a different order:

$$LU = \tilde{A} := \begin{bmatrix} \text{row } p_1 \text{ of } A \\ \vdots \\ \text{row } p_n \text{ of } A \end{bmatrix}$$

The reason for this is the following: If we applied Gaussian elimination *without pivoting* to the matrix $\tilde{A}$, we would get the same L and U that we got for the matrix A *with pivoting*.

Solving a linear system using Gaussian elimination with pivoting: In order to solve a linear system $Ax = b$ we first apply Gaussian elimination with pivoting to the matrix A , yielding L, U, p . Then we know that $LU = \tilde{A}$.

By reordering the equations of $Ax = b$ in the order $p_1, \ldots, p_n$ we obtain the linear system

$$\underbrace{\begin{bmatrix} \text{row } p_1 \text{ of } A \\ \vdots \\ \text{row } p_n \text{ of } A \end{bmatrix}}_{\tilde{A}=LU} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \underbrace{\begin{bmatrix} b_{p_1} \\ \vdots \\ b_{p_n} \end{bmatrix}}_{\tilde{b}}$$

i.e., we have $L(Ux) = \tilde{b}$. We set $y = Ux$. Then we solve the linear system $Ly = \tilde{b}$ using forward substitution, and finally we solve the linear system $Ux = y$ using back substitution.

Summary:

1. Apply Gaussian elimination with pivoting to the matrix A , yielding L , U , p such that $LU = \begin{bmatrix} \text{row } p_1 \text{ of } A \\ \vdots \\ \text{row } p_n \text{ of } A \end{bmatrix}$.
2. Solve $Ly = \begin{bmatrix} b_{p_1} \\ \vdots \\ b_{p_n} \end{bmatrix}$ using forward substitution.
3. Solve $Ux = y$ using back substitution.

Note that this algorithm is also known as Gaussian elimination with *partial* pivoting (partial means that we perform row interchanges, but no column interchanges).

Example: Solve $Ax = b$ for $A = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 1 & 3 \\ 2 & -2 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 2 \\ 1 \\ -4 \end{bmatrix}$. With L and p from the Gaussian elimination the linear system $Ly = (b_{p_1}, b_{p_2, b_{p_3}})^\top = (b_1, b_3, b_2)^\top$ is

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}$$

yielding $y = (2, -5, 2)$ using forward substitution. Then we solve $Ux = y$

$$\begin{bmatrix} 4 & -2 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -5 \\ 2 \end{bmatrix}$$

using back substitution and obtain $x_3 = \frac{1}{2}$, $x_2 = \frac{11}{2}$, $x_3 = \frac{1}{2}$.

Existence and Uniqueness of Solutions

If we perform Gaussian elimination with pivoting on a matrix A (in exact arithmetic), there are two possibilities:

1. The algorithm can be performed without an error message. In this case the diagonal elements $u_{11}, \ldots, u_{nn}$ are all nonzero. For an arbitrary right hand side vector b we can then perform forward substitution (since the diagonal elements of L are 1), and back substitution, without having to divide by zero. Therefore this yields **exactly one solution** x for our linear system.
2. The algorithm stops with an error message. E.g., assume that the algorithm stops in column $j = 3$ since all pivot candidates are zero. This means that a linear system $Ax = b$ is equivalent to the linear system of the form

$$\begin{bmatrix} \circledast & * & * & * & \cdots & * \\ 0 & \circledast & * & * & \cdots & * \\ \vdots & 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & * & \cdots & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix}$$

   Here $*$ denotes an arbitrary number, and $\circledast$ denotes a nonzero number. Let us first consider equations 3 through n of this system. These equations only contain $x_4 \ldots, x_n$. There are two possibilities:

   (a) There is a solution $x_4, \ldots, x_n$ of equations 3 through n . Then we can choose x_3 arbitrarily. Now we can use equation 2 to find x_2 , and finally equation 1 to find x_1 (note that the diagonal elements are nonzero). Therefore we have **infinitely many solutions** for our linear system.

   (b) There is no solution $x_4, \ldots, x_n$ of equations 3 through n . That means that our original linear system has **no solution**.

Observation: In **case 1**. the linear system has a unique solution for any b . In particular, $Ax = 0$ implies $x = 0$. Hence A is **nonsingular**.
In **case 2** we can construct a nonzero solution x for the linear system $Ax = 0$. Hence A is **singular**.

We have shown:

Theorem: If a square matrix A is **nonsingular**, then the linear system $Ax = b$ has a **unique solution** x for any right hand side vector x .
If the matrix A is **singular**, then the linear system $Ax = b$ has either **infinitely many solutions**, or it has **no solution**, depending on the right hand side vector.

Example: Consider $A = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}$. The first step of Gaussian elimination gives $l_{21} = \frac{-2}{4}$ and $U = \begin{bmatrix} 4 & -2 \\ 0 & 0 \end{bmatrix}$. Now we have $u_{22} = 0$, therefore the algorithm stops. Therefore A is singular.

Consider the linear system $\begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$.

By the first step of elimination this becomes $\begin{bmatrix} 4 & -2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$. Note that the second equation has no solution, and therefore the linear system has no solution.

Now consider $\begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$.

By the first step of elimination this becomes $\begin{bmatrix} 4 & -2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$. Note that the second equation is always true. We can choose x_2 arbitrarily, and then determine x_1 from equation 1: $4x_1 - 2x_2 = 2$, yielding $x_1 = (2 + 2x_2)/4$. This gives infinitely many solutions.

Gaussian Elimination with Pivoting in Machine Arithmetic

For a numerical computation we can only expect to find a reasonable answer if the original problem has a unique solution. For a linear system $Ax = b$ this means that the matrix A should be nonsingular. In this case we have found that Gaussian elimination with pivoting, together with forward and back substitution always gives us the answer, at least in exact arithmetic.

It turns out that Gaussian elimination with pivoting can give us solutions with an unnecessarily large roundoff error, depending on the choice of the pivots.

Example We want to solve the following linear system using Gaussian elimination with pivoting:

$$\begin{bmatrix} 4 & -2 & 2 \\ -2 & 1.01 & 3 \\ 2 & -2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Pivoting strategy 1: always select the **first nonzero candidate** as pivot.

Column 1: Pivot selection: $\begin{bmatrix} \textcircled{4} & -2 & 2 \\ -2 & 1.01 & 3 \\ 2 & -2 & 2 \end{bmatrix}$

Elimination: With multipliers $-\frac{1}{2}, \frac{1}{2}$ we obtain

$$\begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & .01 & 4 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 4 \end{bmatrix}$$

Column 2: Pivot selection: $\begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & \boxed{.01} & 4 \\ 0 & -1 & 1 \end{bmatrix}$

Elimination: With multiplier -100 we obtain

$$\begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & \boxed{.01} & 4 \\ 0 & 0 & 401 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 704 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{704}{401}, \qquad x_2 = \frac{7-4x_3}{.01} = \frac{7-4\cdot\frac{704}{401}}{.01}, \qquad x_1 = \frac{4+2x_2-2x_3}{4}$$

In machine arithmetic we will obtain a value $\hat{x}_3$ with relative error of order $\varepsilon_M \approx 10^{-16}$.

Note that $4\cdot\frac{704}{401} \approx 7.022$, hence computing $7-7.022$ causes subtractive cancelation, with magnification factor $\left|\frac{7}{7-7.022}\right| \approx 312$. In machine arithmetic we will therefore obtain a value $\hat{x}_2$ with relative error of order $312\cdot 10^{-16} \approx 3\cdot 10^{-14}$.

In the computation of x_1 there is no subtractive cancelation. But since we are using the value $\hat{x}_2$, we will obtain a value $\hat{x}_1$ with a relative error of order 10^{-14}.

Result: We obtain $\hat{x}_1$ and $\hat{x}_2$ with a relative error of order 10^{-14}, and $\hat{x}_3$ with a relative error of order 10^{-16}.

Pivoting strategy 2: always select the **pivot candidate with the largest absolute value**.

Column 1: same as above

Column 2: Pivot selection: $\begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & .01 & 4 \\ 0 & \boxed{-1} & 1 \end{bmatrix}$, so we interchange rows 2 and 3:

$$\begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & \boxed{-1} & 1 \\ 0 & .01 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 7 \end{bmatrix}$$

Elimination: With multiplier $-.01$ we obtain

$$\begin{bmatrix} \textcircled{4} & -2 & 2 \\ 0 & \boxed{-1} & 1 \\ 0 & 0 & 4.01 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 7.04 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{7.04}{4.01}, \qquad x_2 = \frac{4-x_3}{-1} = \frac{4-\frac{704}{401}}{.01}, \qquad x_1 = \frac{4+2x_2-2x_3}{4}$$

For x_2 we now have to compute $4-\frac{704}{401} \approx 4-1.76$ so there is no subtractive cancelation.

Result: We obtain $\hat{x}_1, \hat{x}_2, \hat{x}_3$ with a relative error of order 10^{-16}.

Conclusion: The first algorithm is numerically unstable, the second algorithm is numerically stable.

This example is typical: Choosing very small pivot elements leads to subtractive cancellation during back substitution when we compute

$$x_j = \frac{y_j - (u_{j,j+1}x_{j+1} + \cdots + u_{j,n}x_n)}{u_{jj}}$$

If x_j has a size of roughly 1 , this means that $y_j - (u_{j,j+1}x_{j+1} + \cdots + u_{j,n}x_n)$ must be of the same size as u_{jj}, hence there will be subtractive cancelation in the subtraction.

Therefore we should use a **pivoting strategy** which avoids small pivot elements. The simplest way to do this is the following: **Select the pivot candidate with the largest absolute value.**

In most practical cases this leads to a numerically stable algorithm, i.e., no unnecessary magnification of roundoff error.

There are very few cases where this algorithm is numerically unstable. We will explain later how to detect and fix this problem.

The inverse matrix

Assume $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix. Then the linear system $Ax = b$ has a unique solution for every $b \in \mathbb{R}^n$. Let $e^{(j)}$ denote the jth unit vector with $e_i^{(j)} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}$, and let $v^{(j)} \in \mathbb{R}^n$ denote the solution of the linear system $Av^{(j)} = e^{(j)}$. Then the $n\times n$ matrix $\left[v^{(1)},\dots,v^{(n)}\right]$ is the **inverse matrix** A^{-1}.

- we have $A^{-1}A = AA^{-1} = I$ with the indentity matrix $I := \left[e^{(1)},\dots,e^{(n)}\right]$
- the solution of the linear system $Ax = b$ is given by $x = A^{-1}b$
- how to compute the inverse matrix A^{-1}:
  - use Gaussian elimination to find L,U,p.
  - use this to solve the linear system for the n right-hand side vectors $\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix},\dots,\begin{bmatrix}0\\\vdots\\0\\1\end{bmatrix}$, yielding the n solution vectors $v^{(1)},\dots,v^{(n)}$
  - let $A^{-1} = \left[v^{(1)},\dots,v^{(n)}\right]$

In **Matlab** we can obtain the inverse matrix as `inv(A)` . **Note that in most applications there is actually no need to find the inverse matrix**.

If we need to compute $x = A^{-1}b$ for a vector b we should use `x=A\b`
If we need to compute $X = A^{-1}M$ for a matrix M we should use `X=A\M`

If we need to compute $x = A^{-1}b$ for several vectors b we should use `[L,U,p]=lu(A,'vector')` and then use `x=U\(L\b(p))` for each vector b.

If we use `inv(A)` then Matlab has first to find the LU-decomposition, and then use this to find the vectors $v^{(1)},\dots,v^{(n)}$ which is a substantial extra work. If the size n is small this does not really matter. But in many applications we have $n > 1000$, and then using `inv(A)` wastes time and gives less accurate results.

Number of operations for numerical computations

When we perform elimination we update elements of the matrix U by subtracting multiples of the pivot row. So we have to perform updates like

$$u_{42} := u_{42} - \ell_{42}\cdot u_{22}$$

On a computer this involves the following operations

- **memory access**: getting $\ell_{42}, u_{22}, u_{42}$ from main memory into the processor at the beginning, writing the new value of u_{42} to main memory at the end
- **multiplication** $t := \ell_{42}\cdot u_{22}$
- **addition/subtraction** $u_{42} := u_{42} - t$

To simplify our bookkeeping, we will **only count multiplications and divisions**. Typically there will be an equal number of additions and subtractions, and memory access operations. We only want to get some rough idea how the work increases depending on the size n of the linear system.

- **finding** L,U,p costs $\boxed{\frac{1}{3}n^3 + O(n^2) \text{ operations}}$
  elimination of column $1,2,\dots,n-1$ costs $n(n-1)+(n-1)(n-2)+\cdots+2\cdot 1 = \frac{1}{3}n^3 + O(n^2)$ operations

- **solving $Ax = b$ if we know L, U, p:** costs $\boxed{n^2 \text{ operations}}$

  solving $Ly = \begin{bmatrix} b_{p_1} \\ \vdots \\ b_{p_n} \end{bmatrix}$: finding $y_1, y_2, \ldots, y_n$ costs $0 + 1 + \cdots + (n-1) = \frac{n(n-1)}{2}$ operations

  solving $Ux = y$: finding $x_n, x_{n-1}, \ldots, x_1$ costs $1 + 2 + \cdots + n = \frac{(n+1)n}{2}$ operations

- **finding A^{-1} if we know L, U, p** costs $\boxed{\frac{2}{3}n^3 + O(n^2) \text{ operations}}$

  finding column $v^{(1)}$: $\frac{n(n-1)}{2}$ for forward substitution, $\frac{(n+1)n}{2}$ for back substitution
  finding column $v^{(2)}$: $\frac{(n-1)(n-2)}{2}$ for forward substitution, $\frac{(n+1)n}{2}$ for back substitution
  $\vdots$
  total: $\frac{1}{6}n^3 + O(n^2)$ for the forward substitutions, $n\frac{(n+1)n}{2}$ for the back substitutions

- **solving $Ax = b$ if we know A^{-1}** costs $\boxed{n^2 \text{ operations}}$

  if we have A^{-1}, finding the matrix-vector product $A^{-1}b$ takes n^2 operations

Comparison

In many applications we have to solve several linear systems with the same matrix A.
We have two possible strategies and the following costs:

	setup for matrix A	for each vector b
Strategy 1	`[L,U,p]=lu(A,'vector')` $\boxed{\frac{1}{3}n^3 + O(n^2)}$	`x=U\(L\b(p))` $\boxed{n^2}$
Strategy 2	`Ai=inv(A)` $\boxed{n^3 + O(n^2)}$	`x=Ai*b` $\boxed{n^2}$

Observations:

- The setup for the matrix A takes most of the work. The additional work for each vector b is very low in comparison.
- Strategy 2 takes about three times as long as Strategy 1