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 $Ax = 0$.

Note that Ax is a linear combination of the column vectors: $Ax = x_1 \cdot (\text{col. } 1) + x_2 \cdot (\text{col. } 2) + x_3 \cdot (\text{col. } 3)$. Therefore: a matrix is singular if the columns are linearly dependent. A matrix is nonsingular if the columns are linearly independent.

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:
   $\mathrm{eq}_2 := \mathrm{eq}_2 - \ell_{21} \cdot \mathrm{eq}_1$, … , $\mathrm{eq}_n := \mathrm{eq}_n - \ell_{n1} \cdot \mathrm{eq}_1$

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

   $\vdots$

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

2. **Back substitution:**
   Solve equation n for x_n .
   Solve equation $n-1$ for x_{n-1} .
   $\vdots$
   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 of three equations (eq1), (eq2), (eq3)

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

Elimination: To eliminate x_1 from equation 2 we let $l_{21} = 4/2 = 2$ and subtract l_{21} times equation 1 from equation 2. To eliminate x_1 from equation 3 we choose $l_{31} = -2/2 = -1$ and subtract l_{31} times equation 1 from equation 3. Now the linear system consists of the three equations $(\mathrm{eq1}') := (\mathrm{eq1})$, $(\mathrm{eq2}') := (\mathrm{eq2}) - l_{21}(\mathrm{eq1})$, $(\mathrm{eq3}') := (\mathrm{eq3}) - l_{31}(\mathrm{eq1})$:

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

To eliminate x_2 from equation 3 we let $l_{32} = 5/(-3)$ and subtract l_{32} times equation 2 from equation 3. Now we have the three equations $(\mathrm{eq1}'') := (\mathrm{eq1}')$, $(\mathrm{eq2}'') := (\mathrm{eq2}')$, $(\mathrm{eq3}'') := (\mathrm{eq3}') - l_{32}(\mathrm{eq2}')$:

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

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). The entries on the diagonal (marked by circles) are called **pivots**. During the algorithm we divide by these numbers, so they need to be nonzero.

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 $x_1 = -\frac{3}{2}$.

Equivalence of the linear systems: We started with the linear system $Ax = b$ consisting of the three equations (eq1), (eq2), (eq3). We ended up with the linear system $Ux = y$ consisting of the three equations (eq1″), (eq2″), (eq3″). Any solution x of $Ax = b$ must be a solution of $Ux = y$, i.e.

$$Ax = b \implies Ux = y$$

since we obtained (eq1′), (eq2′), (eq3′) and then (eq1″), (eq2″), (eq3″) using

$$\begin{array}{lll} (\text{eq}1') := (\text{eq}1), & (\text{eq}2') := (\text{eq}2) - l_{21}(\text{eq}1), & (\text{eq}3') := (\text{eq}3) - l_{31}(\text{eq}1) \\ (\text{eq}1'') := (\text{eq}1'), & (\text{eq}2'') := (\text{eq}2') & (\text{eq}3'') := (\text{eq}3') - l_{32}(\text{eq}2') \end{array}$$

We claim that any solution x of $Ux = y$ must also be a solution of $Ax = b$, i.e.,

$$Ux = y \implies Ax = b$$

Proof: Assume that (eq1″), (eq2″), (eq3″) hold. Then we can get back to (eq1′), (eq2′), (eq3′) and finally to (eq1), (eq2), (eq3) as follows:

$$\begin{array}{lll} (\text{eq}1') := (\text{eq}1''), & (\text{eq}2') := (\text{eq}2''), & (\text{eq}3') := (\text{eq}3'') + l_{32}(\text{eq}2'') \\ (\text{eq}1) := (\text{eq}1'), & (\text{eq}2) := (\text{eq}2') + l_{21}(\text{eq}1'), & (\text{eq}3) := (\text{eq}3') + l_{31}(\text{eq}1') \end{array}$$

Note that we can write this as

$$(\text{eq}1) := (\text{eq}1''), \qquad (\text{eq}2) := (\text{eq}2'') + l_{21}(\text{eq}1''), \qquad (\text{eq}3) := (\text{eq}3'') + l_{31}(\text{eq}1'') + l_{32}(\text{eq}2'')$$

Therefore the **right hand side vectors** b **and** y are related as follows

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

and the **coefficient matrices** A **and** U are related as follows

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

Result: We define the lower triangular matrix $L := \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}$ containing the multipliers. Then the process of getting from (eq1″), (eq2″), (eq3″) to the original equations (eq1), (eq2), (eq3) is described by

$$\boxed{b = Ly, \qquad A = LU}$$

The equation $A = LU$ is called **LU decomposition**.

Summary: Now we can rephrase the **algorithm for solving a linear system** $Ax = b$ as follows:

1. **Perform Gaussian elimination on the matrix** A, yielding 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 :

$$L := \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n,1} & \cdots & \ell_{n,n-1} & 1 \end{bmatrix}, \qquad U := \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix},$$

2. **Transform the right hand side vector b to the vector y:** Solve $Ly = b$ using forward substitution

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n,1} & \cdots & \ell_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

3. **Solve $Ux = y$ using back substitution**

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

Finding the LU decomposition takes the most work. It needs many more operations than performing forward or back substitution.

Example:

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

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

step 1:

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

step 2:

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

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

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, yielding $x_3 = 1, x_2 = 1, x_1 = -\frac{3}{2}$.

Solving several linear systems with the same matrix A: Often we have to solve several linear systems with different right hand side vectors $b, \tilde{b} \in \mathbb{R}^n$. We want to find the solution x of $Ax = b$ and the solution $\tilde{x}$ of $A\tilde{x} = \tilde{b}$. We proceed as follows:

- Use Gaussian elimination on the matrix A to find the LU decomposition $A = LU$.
- For the first right hand side vector b: Solve $Ly = b$ using forward substitution, solve $Ux = y$ using back substitution.
- For the second right hand side vector $\tilde{b}$: Solve $L\tilde{y} = \tilde{b}$ using forward substitution, solve $U\tilde{x} = \tilde{y}$ using back substitution.

Note that the Gaussian elimination of the matrix is only performed once (this is the part which takes the most work). We store the matrices L and U, and can then solve any linear system very easily using forward and back substitution.

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 **permuatation vector** p which is initially $(1, 2, \ldots, n)^\top$, and we **interchange the rows of** p whenever we interchange the rows of U.

Pivoting for column j:

- Look at the **pivot candidates** $u_{j,j}, u_{j+1,j}, \ldots, u_{n,j}$.
  Pick a nonzero pivot candidate $u_{k,j}$.
  If *all* pivot candidates are zero: Stop the algorithm, the matrix is singular.
- Interchange rows j and k of L, U, p

Example:

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

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

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

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

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

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

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

How do we use L, U, p to solve a linear system $Ax = b$?

Gaussian elimination with pivoting gives a vector p containing the numbers $1, \dots, n$ in a permuted order. Our linear system $Ax = b$ consists of $(\text{eq}1), \dots, (\text{eq}n)$. We can reorder these equations and write them down in the order $(\text{eq}p_1), \dots (\text{eq}p_n)$, this gives the following linear system $\tilde{A}x = \tilde{b}$:

$$\underbrace{\begin{bmatrix} (\text{row } p_1 \text{ of} A) \\ \vdots \\ (\text{row } p_1 \text{ of} A) \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \underbrace{\begin{bmatrix} b_{p_1} \\ \vdots \\ b_{p_n} \end{bmatrix}}_{\tilde{b}} \tag{2}$$

What happens if we **perform Gaussian elimination without pivoting on the matrix** $\tilde{A}$? In our example we have $\tilde{A} = \begin{bmatrix} (\text{row } 1 \text{ of} A) \\ (\text{row } 3 \text{ of} A) \\ (\text{row } 2 \text{ of} A) \end{bmatrix} = \begin{bmatrix} 4 & -2 & 2 \\ 2 & -2 & 2 \\ -2 & 1 & 3 \end{bmatrix}$ and elimination in column 1 gives

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

Note that now the correct pivot -1 is already in the correct position in column 2. Hence Gaussian elimination **without** pivoting for the matrix $\tilde{A}$ gives the same matrices $L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix}$, $U = \begin{bmatrix} 4 & -2 & 2 \\ 0 & -1 & 1 \\ 0 & 0 & 4 \end{bmatrix}$ we got from Gaussian elimination with pivoting for the matrix A.

This is true in general: Let p be the permutation vector from Gaussian elimination with pivoting for the matrix A, and $\tilde{A} := \begin{bmatrix} (\text{row } p_1 \text{ of} A) \\ \vdots \\ (\text{row } p_1 \text{ of} A) \end{bmatrix}$. Then Gaussian elimination without pivoting for the matrix $\tilde{A}$ has in each column the correct pivot already in the correct position, and we get the same matrices L and U we got from Gaussian elimination **with** pivoting for the matrix A.

Hence:

- The original linear system $Ax = b$ is equivalent to $\tilde{A}x = \tilde{b}$
- Gaussian elimination without pivoting for $\tilde{A}$ gives the same L, U
- Hence we can solve the linear system $\tilde{A}x = \tilde{b}$ as follows:
  solve $Ly = \tilde{b}$ by forward substitution
  solve $Ux = y$ by back substitution

Result: Algorithm for solving a linear system $Ax = b$

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**.

This algorithm is also known as **Gaussian elimination with *partial* pivoting.** Here "partial" means that we perform row interchanges, but no column interchanges. (In contrast, Gaussian elimination with *full* pivoting uses row and column interchanges.)

Example: Solve $Ax=b$ for $A=\begin{bmatrix}4&-2&2\\-2&1&3\\2&-2&2\end{bmatrix}$, $b=\begin{bmatrix}2\\1\\-4\end{bmatrix}$.

Gaussian elimination with pivoting gives $L=\begin{bmatrix}1&0&0\\\frac{1}{2}&1&0\\-\frac{1}{2}&0&1\end{bmatrix}$, $U=\begin{bmatrix}4&-2&2\\0&-1&1\\0&0&4\end{bmatrix}$, $p=\begin{bmatrix}1\\3\\2\end{bmatrix}$ (see above).

Forward substitution: The linear system $Ly=\begin{bmatrix}b_{p_1}\\b_{p_2}\\b_{p_3}\end{bmatrix}=\begin{bmatrix}b_1\\b_3\\b_2\end{bmatrix}$ is

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

and forward substitution gives the solution $y=\begin{bmatrix}2\\-5\\2\end{bmatrix}$.

Back substitution: The linear system $Ux=y$ is

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

and back substitution gives the solution $x_3=\frac{1}{2}$, $x_2=\frac{11}{2}$, $x_3=\frac{1}{2}$.

What happens if Gaussian elimination with pivoting breaks down?

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

1. **We obtain nonzero pivots** $u_{11},u_{22},\dots,u_{nn}$**.**
   The linear system $Ax=b$ is equivalent to the linear system $Ux=y$. Since $u_{11},\dots,u_{nn}$ are nonzero, this linear system has a unique solution vector x. Hence the original linear system has a unique solution vector x, i.e., the **matrix** A **is nonsingular**.

2. **The algorithm breaks down in in column** j**.** This means we were able to perform elimination for columns $1,\dots,j-1$, but then **all pivot candidates in column** j **are zero**.
   E.g., assume that the algorithm breaks down in column $j=3$ since all pivot candidates are zero. This means that the linear system $A x=\vec{0}$ is equivalent to the following linear system $Ux=\vec{0}$:

$$\begin{bmatrix}\circledast&*&*&*&\cdots&*\\0&\circledast&*&*&\cdots&*\\\vdots&0&0&*&\cdots&*\\\vdots&\vdots&\vdots&\vdots&&\vdots\\0&0&0&*&\cdots&*\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\\vdots\\x_n\end{bmatrix}=\begin{bmatrix}0\\0\\0\\\vdots\\0\end{bmatrix}\tag{3}$$

   Here $*$ denotes an arbitrary number, and $\circledast$ denotes a nonzero number

   We claim that we can construct a vector $x\neq\vec{0}$ such that $Ax=\vec{0}$. Consider the vector $x=\begin{bmatrix}x_1\\x_2\\1\\0\\\vdots\\0\end{bmatrix}$. This vector already satisfies eq.$3,\dots,$eq.n of (3). We can then determine x_2 and x_1 by back substitution: eq.2 gives a unique x_2 since $u_{22}\neq 0$. Then eq.1 gives a unique x_1 since $u_{11}\neq 0$.
   Since $x\neq\vec{0}$ and $Ax=\vec{0}$ the **matrix** A **is singular**.

How to solve a linear system in Matlab

Solving a linear system $Mx = b$ where the matrix $M \in \mathbb{R}^{n\times n}$ is **upper or lower triangular**: Use the Matlab command **x=M\b** Matlab uses the appropriate version of back or forward substitution. (This actually works if M is any row permutation of an upper triangular matrix.)

Row permutation: The vector p contains a permutation of the numbers $1,\dots,n$. For a vector $b \in \mathbb{R}^n$ the vector $\begin{bmatrix} b_{p_1} \\ \vdots \\ b_{p_n} \end{bmatrix}$ is given in Matlab by **b(p)**

For a matrix $B \in \mathbb{R}^{n\times k}$ the matrix $\begin{bmatrix} \text{row } p_1 \text{ of } B \\ \vdots \\ \text{row } p_n \text{ of } B \end{bmatrix}$ is given in Matlab by **B(p,:)**

Solving a linear system $Ax = b$ in Matlab:

```
[L,U,p] = lu(A,'vector'); % Gaussian elimination with pivoting, return p as a vector
y = L\b(p);               % forward substitution
x = U\y;                  % back substitution
```

We can also use the shortcut **x=A\b** which executes the three commands above.

Solving several linear systems with the same matrix in Matlab:

We want solve the two linear systems $Ax = b$ and $A\hat{x} = \hat{b}$. We could use **x=A\b; xh=A\bh;**

But this performs the Gaussian elimination of the matrix twice, and this is the part of the algorithm which takes the most work.

Therefore we should rather use

```
[L,U,p] = lu(A,'vector'); % use Gaussian elimination with pivoting to find L,U,p
x = U\(L\b(p));           % use L,U,p to solve A x = b
xh = U\(L\bh(p));         % use L,U,p to solve A xh = bh
```

Example for Gaussian Elimination with Pivoting with $A \in \mathbb{R}^{4\times 4}$

Solve the linear system $\begin{bmatrix} 0 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 2 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 4 \end{bmatrix}$. Use the **pivot candidate with the largest absolute value.**

Gaussian Elimination for matrix A:

Initialize L, U, p. Then select pivot for column 1:

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

Move pivot for column 1 in position: interchange rows 1 and 4

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

Elimination in column 1. Then select pivot for column 2

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{5}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 & 1 \end{bmatrix}, \qquad p = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

Move pivot for column 2 in position: interchange rows 2 and 3

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & \frac{5}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{3}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 & 1 \end{bmatrix}, \qquad p = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Elimination in column 2. Then select pivot for column 3

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{3}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & \frac{5}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & \frac{1}{5} & \frac{4}{5} \\ 0 & 0 & 1 & 1 \end{bmatrix}, \qquad p = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Move pivot for column 3 in position: interchange rows 3 and 4

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{3}{5} & 0 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & \frac{5}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 & 1 \\ 0 & 0 & \frac{1}{5} & \frac{4}{5} \end{bmatrix}, \qquad p = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 2 \end{bmatrix}$$

Elimination in column 3

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{3}{5} & \frac{1}{5} & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & \frac{5}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \frac{3}{5} \end{bmatrix}, \qquad p = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 2 \end{bmatrix}$$

The last pivot is $\frac{3}{5}$. This is nonzero, so the algorithm succeeded. Therefore A is **nonsingular**.

Finally put 1's on the diagonal of L, yielding $L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & \frac{3}{5} & \frac{1}{5} & 1 \end{bmatrix}$.

Given b, use L, U, p to solve linear system:

Solve $Ly = \begin{bmatrix} b_{p_1} \\ \vdots \\ b_{p_n} \end{bmatrix}$ by forward substitution:

$$\text{Solving} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & \frac{3}{5} & \frac{1}{5} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 0 \\ 1 \end{bmatrix} \quad \text{gives} \quad y = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$

Solve $Ux = y$ by back substitution:

$$\text{Solving} \quad \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & \frac{5}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \frac{3}{5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 3 \end{bmatrix} \quad \text{gives} \quad x = \begin{bmatrix} 1 \\ 2 \\ -5 \\ 5 \end{bmatrix}$$

How to do this in Matlab:

```
>> A = [0 0 1 1;-1 1 0 0; 1 3 1 0; 2 1 1 1]
A =
     0     0     1     1
    -1     1     0     0
     1     3     1     0
     2     1     1     1
>> [L,U,p] = lu(A,'vector')
L =
            1            0            0            0
          0.5            1            0            0
            0            0            1            0
         -0.5          0.6          0.2            1
U =
            2            1            1            1
            0          2.5          0.5         -0.5
            0            0            1            1
            0            0            0          0.6
p =
     4     3     1     2
>> b = [0;1;2;4];
>> y = L\b(p)
y =
     4
     0
     0
     3
>> x = U\y
x =
            1
            2
           -5
            5
```

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: $\begin{bmatrix} 4 & -2 & 2 \\ -2 & 1.01 & 3 \\ 2 & -2 & 2 \end{bmatrix}$

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

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

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

Elimination: With multiplier -100 we obtain

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

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

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

Elimination: With multiplier $-.01$ we obtain

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

Back substitution:

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

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 $\left[v^{(1)}, \ldots, v^{(n)}\right]$ is the **inverse matrix** A^{-1}.

- we have $A^{-1}A = AA^{-1} = I$ with the indentity matrix $I := \left[e^{(1)}, \ldots, e^{(n)}\right]$
- 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 $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$, yielding the n solution vectors $v^{(1)}, \ldots, v^{(n)}$
  - let $A^{-1} = \left[v^{(1)}, \ldots, v^{(n)}\right]$

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 $b \in \mathbb{R}^n$ we should use **`x=A\b`**
If we need to compute $X = A^{-1}B$ for a matrix $B \in \mathbb{R}^{n\times k}$ we should use **`X=A\B`**

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.
We can compute $X = A^{-1}B$ for a matrix B using **`X=U\(L\B(p,:))`**

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 $y_1, y_2, \dots, y_n$ costs $0 + 1 + \cdots + (n-1) = \frac{n(n-1)}{2}$ operations
  solving $Ux = y$: finding $x_n, x_{n-1}, \dots, x_1$ 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')`	`x=U\(L\b(p))`
	$\boxed{\frac{1}{3}n^3 + O(n^2)}$	$\boxed{n^2}$
Strategy 2	`Ai=inv(A)`	`x=Ai*b`
	$\boxed{n^3 + O(n^2)}$	$\boxed{n^2}$

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