Introduction to Linear Algebra

1. Vectors and Matrices
   - matrix notation
   - solving triangular linear systems

2. Gaussian Elimination
   - row reduction to upper triangular form
   - LU factorization
   - the Doolittle algorithm

MCS 471 Lecture 8
Numerical Analysis
Jan Verschelde, 9 September 2022

Introduction to Linear Algebra

1. Vectors and Matrices
   - matrix notation
   - solving triangular linear systems

2. Gaussian Elimination
   - row reduction to upper triangular form
   - LU factorization
   - the Doolittle algorithm

Matrices and Vectors

A real n-by-m matrix $A \in \mathbb{R}^{n \times m}$ has n rows and m columns:

$$A = [a_{i,j}] = \left[\begin{array}{cccc} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{array} \right], \quad \begin{array}{l} a_{i,j} \in \mathbb{R}, \\ i = 1, 2, \ldots, n, \\ j = 1, 2, \ldots, m. \end{array}$$

Vectors are by default column vectors, n-by-1 matrices, $\mathbf{b} \in \mathbb{R}^n$:

$$\mathbf{b} = \left[\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_n \end{array} \right], \quad b_i \in \mathbb{R}.$$

Matrix Operations

Operations on matrices:

- Scalar multiplication: $r \cdot A = [r \cdot a_{i,j}]$, $r \in \mathbb{R}$,
  is a componentwise operation, done on every component of A.
- Addition/subtraction of two matrices:

$$A, B \in \mathbb{R}^{n \times m} : A = [a_{i,j}], B = [b_{i,j}] : A \pm B = [a_{i,j} \pm b_{i,j}].$$

  A and B must have the same number of rows and columns.
- Multiplication of $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{m \times p}$: $C = A \star B \in \mathbb{R}^{n \times p}$.

$$C = [c_{i,j}], \quad c_{i,j} = \sum_{k=1}^{m} a_{i,k} \cdot b_{k,j}.$$

  The number of columns of A must equal the number of rows of B.

properties of matrix multiplication

The product of two square matrices does not commute:
There are $A, B \in \mathbb{R}^{n \times n}$: $A \star B \neq B \star A$.

Exercise 1: Construct two 2-by-2 matrices A and B so that

$$A \star B \neq B \star A.$$

A matrix $A = [a_{i,j}]$ is *upper triangular* if all elements below the diagonal are zero, i.e.: $a_{i,j} = 0$ for all $i > j$.

Exercise 2: Show that the product of two upper triangular matrices is again an upper triangular matrix.

Matrix-Vector Multiplication

A linear system of n equations in m unknowns:

$$\left\{ \begin{array}{rcl} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,m}x_m & = & b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,m}x_m & = & b_2 \\ & \vdots & \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,m}x_m & = & b_n \end{array} \right.$$

is written in matrix form as

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

and abbreviated in matrix notation as

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{n \times m}, \quad \mathbf{x} \in \mathbb{R}^m, \quad \mathbf{b} \in \mathbb{R}^n.$$

Introduction to Linear Algebra

1. Vectors and Matrices
   - matrix notation
   - solving triangular linear systems
2. Gaussian Elimination
   - row reduction to upper triangular form
   - LU factorization
   - the Doolittle algorithm

solving upper triangular linear systems

An upper triangular matrix $U \in \mathbb{R}^{n \times n}$, $U = [u_{i,j}]$, $u_{i,i} \neq 0$, for all i, and right handside vector $\mathbf{b}$ defines a linear system $U\mathbf{x} = \mathbf{b}$.

$$\begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \left\{ \begin{array}{rcl} u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 & = & b_1 \\ u_{2,2}x_2 + u_{2,3}x_3 & = & b_2 \\ u_{3,3}x_3 & = & b_3 \end{array} \right.$$

The last equation is an equation in one variable:

$$u_{3,3}x_3 = b_3 \quad \Rightarrow \quad x_3 = b_3/u_{3,3}$$

and we can then solve the second to last equation for x_2:

$$u_{2,2}x_2 + u_{2,3}x_3 = b_2 \quad \Rightarrow \quad x_2 = (b_2 - u_{2,3}x_3)/u_{2,2},$$

and then we solve for x_1:

$$u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 = b_1 \quad \Rightarrow \quad x_1 = (b_1 - u_{1,2}x_2 - u_{1,3}x_3)/u_{1,1}.$$

general back substitution formulas

Assume all $u_{i,i} \neq 0$, for all i, in the linear system

$$\left\{ \begin{array}{rcl} u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 + \cdots + u_{1,m-1}x_{n-1} + u_{1,n}x_n & = & b_1 \\ u_{2,2}x_2 + u_{2,3}x_3 + \cdots + u_{2,n-1}x_{n-1} + u_{2,n}x_n & = & b_2 \\ & \vdots & \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n & = & b_{n-1} \\ u_{n,n}x_n & = & b_n \end{array} \right.$$

then the formulas below solve the linear system:

$$\begin{array}{rcl} x_n & = & b_n / u_{n,n} \\ x_{n-1} & = & (b_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1} \\ & \vdots & \\ x_2 & = & (b_2 - u_{2,3}x_3 - \cdots - u_{2,n}x_n)/u_{2,2} \\ x_1 & = & (b_1 - u_{1,2}x_2 - u_{1,3}x_3 - \cdots - u_{1,n}x_n)/u_{1,1}. \end{array}$$

the back substitution algorithm

$$\begin{array}{rcl} x_n & = & b_n / u_{n,n} \\ x_{n-1} & = & (b_{n-1} - u_{n-1,n} x_n) / u_{n-1,n-1} \\ & \vdots & \\ x_2 & = & (b_2 - u_{2,3} x_3 - \cdots - u_{2,n} x_n) / u_{2,2} \\ x_1 & = & (b_1 - u_{1,2} x_2 - \cdots - u_{1,n} x_n) / u_{1,1}. \end{array}$$

Input: $U \in \mathbb{R}^{n \times n}$, $U = [u_{i,j}]$, $u_{i,i} \neq 0$, $i = 1, 2, \ldots, n$;
$\mathbf{b} \in \mathbb{R}^n$, a right hand side vector.
Output: $\mathbf{x} \in \mathbb{R}^n$ so that $U\mathbf{x} = \mathbf{b}$.

for $i = n, n-1, \ldots, 1$ do
  $x_i := b_i$
  for $j = i+1, i+2, \ldots, n$ do
    $x_i := x_i - u_{i,j} \star x_j$
  $x_i := x_i / u_{i,i}$.

cost of the back substitution algorithm

for $i = n, n-1, \ldots, 1$ do
    $x_i := b_i$
    for $j = i+1, i+2, \ldots, n$ do
        $x_i := x_i - u_{i,j} \star x_j$
    $x_i := x_i / u_{i,i}$.

We count the number of operations of the innermost loop:

$$
\begin{array}{rrcl}
i = n-1, & j = n & : & 1 \\
i = n-2, & j = n-1, n & : & 2 \\
 & & \vdots & \\
i = 2, & j = 3, 4, \ldots, n & : & n-2 \\
i = 1, & j = 2, 3, \ldots, n & : & n-1 \\
\hline
\multicolumn{2}{r}{1 + 2 + \cdots + (n-2) + (n-1)} & = & n(n-1)/2
\end{array}
$$

the cost is quadratic in the dimension

The cost to solve a linear system of dimension n by back substitution:

- $n(n-1)/2$ subtractions,
- $n(n-1)/2$ multiplications, and
- n divisions.

We have thus proven the following theorem.

Theorem

The cost to solve an upper triangular system of dimension n is $O(n^2)$.

forward substitution to solve lower triangular systems

A lower triangular system is defined by a lower triangular matrix L, $L = [\ell_{i,j}]$, where all elements above the diagonal are zero: $\ell_{i,j} = 0$ for all $j > i$, and a right hand side vector $\mathbf{b} \in \mathbb{R}^n$.

$$\left\{ \begin{array}{rcl} \ell_{1,1}x_1 & = & b_1 \\ \ell_{2,1}x_1 + \ell_{2,2}x_2 & = & b_2 \\ & \vdots & \\ \ell_{n,1}x_1 + \ell_{n,2}x_2 + \cdots + \ell_{n,n}x_n & = & b_n \end{array} \right.$$

Exercise 3: Write formulas to solve a lower triangular linear system.
Exercise 4: Turn the formulas of Exercise 3 into an algorithm.
Exercise 5: Show that the cost to solve a lower triangular linear system of n equations in n variables is $O(n^2)$.

solving a random linear system in Julia

```
julia> A = rand(2,2)
2x2 Array{Float64,2}:
 0.28993     0.756795
 0.00655962  0.382648

julia> b = rand(2,1)
2x1 Array{Float64,2}:
 0.901888
 0.347995

julia> x = A\b
2x1 Array{Float64,2}:
 0.771342
 0.896217

julia> A*x
2x1 Array{Float64,2}:
 0.901888
 0.347995
```

computing the norm

To compute the accuracy of the result `x`,
we compute the residual, or backward error.

For this we need `norm` of the `LinearAlgebra` module.
The session of the previous slide continues ...

```
julia> using LinearAlgebra

julia> norm(b-A*x)
0.0
```

In mathematical notation, we compute $\|\mathbf{b} - A\mathbf{x}\|$.

The $\|\cdot\|$ is *a vector norm* (for more see L-9),
it measures the length of a vector.

Introduction to Linear Algebra

1. Vectors and Matrices
   - matrix notation
   - solving triangular linear systems

2. Gaussian Elimination
   - row reduction to upper triangular form
   - LU factorization
   - the Doolittle algorithm

row reduction

The linear system $A\mathbf{x} = \mathbf{b}$, for $A \in \mathbb{R}^{n \times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^n$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

is reduced to an upper triangular system

$$\begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,m} \\ 0 & u_{2,2} & \cdots & u_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

for an upper triangular matrix $U \in \mathbb{R}^{n \times n}$ and some vector $\mathbf{y} \in \mathbb{R}^n$.

an example

To reduce $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ to an upper triangular matrix, take the second row R_2 and substract from R_2 the first row R_1 times 3.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \xrightarrow{R_2 := R_2 - 3R_1} \quad U = \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$$

The operation $R_2 := R_2 - 3R_1$ can be written as a matrix multiplication

$$M = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix}, \quad M \star A = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix} = U.$$

The inverse of $R_2 := R_2 - 3R_1$ is $R_2 := R_2 + 3R_1$ so we have

$$M^{-1} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \star \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$$

as $M^{-1} \star (M \star A = U)$ implies $A = M^{-1} \star U$.

Introduction to Linear Algebra

1. Vectors and Matrices
   - matrix notation
   - solving triangular linear systems

2. Gaussian Elimination
   - row reduction to upper triangular form
   - LU factorization
   - the Doolittle algorithm

LU factorization

Definition

Given $A \in \mathbb{R}^{n \times n}$, *the LU factorization* of A writes A as $L \star U$, where

- L is a lower triangular n-by-n matrix, and
- U is an upper triangular n-by-n matrix.

Application: solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{b} \in \mathbb{R}^n$.

1. Compute L and U so that $A = L \star U$, then:

$$A = L \star U \text{ implies } A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad (L \star U)\mathbf{x} = \mathbf{b}$$
$$\Leftrightarrow \quad L(U\mathbf{x}) = \mathbf{b}.$$

2. Denote $U\mathbf{x}$ by $\mathbf{y}$ and solve $L\mathbf{y} = \mathbf{b}$.
3. Solve $U\mathbf{x} = \mathbf{y}$.

determinants

Another application of LU factorization is
the computation of the determinant of a matrix A, denoted by $\det(A)$.

We have the following properties of $\det(\cdot)$:

1. For a lower triangular matrix $L = [\ell_{i,j}]$, $\ell_{i,j} = 0$ for all $j > i$:
$$\det(L) = \ell_{1,1} \cdot \ell_{2,2} \cdots \ell_{n,n}.$$
2. For an upper triangular matrix $U = [u_{i,j}]$, $u_{i,j} = 0$ for all $j < i$:
$$\det(U) = u_{1,1} \cdot u_{2,2} \cdots u_{n,n}.$$
3. For a product for two square matrices A and B:
$$C = A \star B \Rightarrow \det(C) = \det(A) \cdot \det(B).$$

For an LU factorization of A, $A = L \star U$, $\det(A) = \det(L) \cdot \det(U)$.

$$\det(A) = \ell_{1,1} \cdot \ell_{2,2} \cdots \ell_{n,n} \cdot u_{1,1} \cdot u_{2,2} \cdots u_{n,n}.$$

For any $\mathbf{b}$, $A\mathbf{x} = \mathbf{b}$ has a unique solution if and only if $\det(A) \neq 0$.

more exercises

Exercise 6: Consider

$$A = \begin{bmatrix} 4 & -2 & 1 \\ -3 & -1 & 4 \\ 1 & -1 & 3 \end{bmatrix}.$$

- Compute the three multiplication matrices needed to reduce A to an upper triangular matrix. Use exact rational arithmetic.
- Shows that the product of the inverses of the multiplication matrices, multiplied in the correct order, gives the L in the LU factorization of A.

Exercise 7: Use the LU factorization of Exercise 6 to compute $\det(A)$.

Introduction to Linear Algebra

deriving the formulas for LU factorization

We have choice and set the diagonal elements of L to one.
Consider a general three dimensional matrix:

$$\left[\begin{array}{ccc} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{array}\right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ \ell_{2,1} & 1 & 0 \\ \ell_{3,1} & \ell_{3,2} & 1 \end{array}\right] \star \left[\begin{array}{ccc} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{array}\right]$$

$$= \left[\begin{array}{ccc} u_{1,1} & u_{1,2} & u_{1,3} \\ \ell_{2,1} u_{1,1} & \ell_{2,1} u_{1,2} + u_{2,2} & \ell_{2,1} u_{1,3} + u_{2,3} \\ \ell_{3,1} u_{1,1} & \ell_{3,1} u_{1,2} + \ell_{3,2} u_{2,2} & \ell_{3,1} u_{1,3} + \ell_{3,2} u_{2,3} + u_{3,3} \end{array}\right]$$

The first column in the product looks simple.
Start with the first column, then move to the second colum.

deriving the formulas for LU factorization continued

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ \ell_{2,1}u_{1,1} & \ell_{2,1}u_{1,2} + u_{2,2} & \ell_{2,1}u_{1,3} + u_{2,3} \\ \ell_{3,1}u_{1,1} & \ell_{3,1}u_{1,2} + \ell_{3,2}u_{2,2} & \ell_{3,1}u_{1,3} + \ell_{3,2}u_{2,3} + u_{3,3} \end{bmatrix}$$

1. $u_{1,1} := a_{1,1}$
   $\ell_{2,1} := a_{2,1}/u_{1,1}$
   $\ell_{3,1} := a_{3,1}/u_{1,1}$
2. $u_{1,2} := a_{1,2}$
   $u_{2,2} := a_{2,2} - \ell_{2,1}u_{1,2}$
   $\ell_{3,2} := (a_{3,2} - \ell_{3,1}u_{1,2})/u_{2,2}$
3. $u_{1,3} := a_{1,3}$
   $u_{2,3} := a_{2,3} - \ell_{2,1}u_{1,3}$
   $u_{3,3} := a_{3,3} - \ell_{3,1}u_{1,3} - \ell_{3,2}u_{2,3}$

the algorithm for the LU factorization

1. $u_{1,1} := a_{1,1}$
   $\ell_{2,1} := a_{2,1}/u_{1,1}$
   $\ell_{3,1} := a_{3,1}/u_{1,1}$
2. $u_{1,2} := a_{1,2}$
   $u_{2,2} := a_{2,2} - \ell_{2,1}u_{1,2}$
   $\ell_{3,2} := (a_{3,2} - \ell_{3,1}u_{1,2})/u_{2,2}$
3. $u_{1,3} := a_{1,3}$
   $u_{2,3} := a_{2,3} - \ell_{2,1}u_{1,3}$
   $u_{3,3} := a_{3,3} - \ell_{3,1}u_{1,3} - \ell_{3,2}u_{2,3}$

for $j = 1, 2, \ldots, n$ do
    for $i = 1, 2, \ldots, j$ do
        $u_{i,j} := a_{i,j}$
        for $k = 1, 2, \ldots, i-1$ do
            $u_{i,j} := u_{i,j} - \ell_{i,k}u_{k,j}$
    for $i = j+1, \ldots, n$ do
        $\ell_{i,j} := a_{i,j}$
        for $k = 1, 2, \ldots, j-1$
            $\ell_{i,j} := \ell_{i,j} - \ell_{i,k}u_{k,j}$
        $\ell_{i,j} := \ell_{i,j}/u_{j,j}$

Note that the L and the U fit in the matrix A,
the numbers $a_{i,j}$ may be replaced by $\ell_{i,j}$ and $u_{i,j}$.

a Julia function

```
"""
Returns the matrices L and U in an LU factorization of A.
"""
function lufac(mat::Array{Float64})
    nbrows, nbcols = size(mat)
    low = eye(nbrows, nbcols)
    upp = zeros(nbrows, nbcols)
    for j=1:nbcols
        for i=1:j
            upp[i,j] = mat[i,j]
            for k=1:i-1
                upp[i,j] = upp[i,j] - low[i,k]*upp[k,j]
            end
        end
        for i=j+1:nbrows
            low[i, j] = mat[i,j]
            for k=1:j-1
                low[i, j] = low[i, j] - low[i, k]*upp[k, j]
            end
            low[i, j] = low[i, j]/upp[j, j]
        end
    end
    return (low, upp)
end
```

the cost of the LU factorization

for $j = 1, 2, \ldots, n$ do
  for $i = 1, 2, \ldots, j$ do
    $u_{i,j} := a_{i,j}$
    for $k = 1, 2, \ldots, i-1$ do
      $u_{i,j} := u_{i,j} - \ell_{i,k} u_{k,j}$
  for $i = j+1, \ldots, n$ do
    $\ell_{i,j} := a_{i,j}$
    for $k = 1, 2, \ldots, j-1$
      $\ell_{i,j} := \ell_{i,j} - \ell_{i,k} u_{k,j}$
    $\ell_{i,j} := \ell_{i,j} / u_{j,j}$

The first loop runs from 1 to n and the inner second loop takes i from 1 to j and then from $j+1$ to n, thus i also runs from 1 to n.
The second inner and third loops take $n(n-1)/2$ steps, following the same arguments used in computing the cost of the back substitution.

the cost of the LU factorization continued

$n(n-1)/2$ multiplied with n gives an $O(n^3)$ cost, so we have:

Theorem

The cost of an LU factorization of a matrix of dimension n is $O(n^3)$.

To solve an n-dimensional system $A\mathbf{x} = \mathbf{b}$, with $A = LU$:

$$(LU)\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad L(U\mathbf{x}) = \mathbf{b} \quad \Rightarrow \quad \text{let } \mathbf{y} = U\mathbf{x} \text{ then } L\mathbf{y} = \mathbf{b}.$$

So $A\mathbf{x} = \mathbf{b}$ is reduced to solving $L\mathbf{y} = \mathbf{b}$ and $U\mathbf{x} = \mathbf{y}$.

1. Compute L and U so that $A = L \star U$ costs $O(n^3)$.
2. Solving $L\mathbf{y} = \mathbf{b}$ is $O(n^2)$.
3. Solving $U\mathbf{x} = \mathbf{y}$ is $O(n^2)$.

Theorem

The cost of solving an n-dimension system is $O(n^3)$.

verification with a Julia session

We make a system that has $\left[\begin{array}{c} 1 \\ 1 \end{array}\right]$ as a solution:

```
julia> using LinearAlgebra

julia> A = [4.0 2.0; 1.0 3.0]; x = ones(2,1); b = A*x;

julia> L, U = lu(A)
LU{Float64,Array{Float64,2}}
L factor:
2×2 Array{Float64,2}:
 1.0   0.0
 0.25  1.0
U factor:
2×2 Array{Float64,2}:
 4.0  2.0
 0.0  2.5
```

the Julia session continued

Given L and U: $A = LU$, we solve $A\mathbf{x} = \mathbf{b}$ in two steps:

1. solve $L\mathbf{y} = \mathbf{b}$
2. solve $U\mathbf{x} = \mathbf{y}$

```
julia> y = L\b
2×1 Array{Float64,2}:
 6.0
 2.5

julia> U\y
2×1 Array{Float64,2}:
 1.0
 1.0
```

computing the inverse

The inverse A^{-1} satisfies $AA^{-1} = I$,
where I is the identity matrix with columns $\mathbf{e}_1$, $\mathbf{e}_2$, ..., $\mathbf{e}_n$.

Let $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ denote the columns of A^{-1}.

Computing A^{-1} requires the solving of n linear systems:

$$A\mathbf{x}_i = \mathbf{e}_i, \quad i = 1, 2, \ldots, n.$$

The LU factorization of A needs to be computed only once,
so the cost of computing A^{-1} is $O(n^3)$.