Row Pivoting for Numerical Stability

1. Motivation
   - the need for pivoting
   - a numerical example
2. PLU Factorization
   - partial pivoting
   - a numerical example
3. Numerical Stability
   - a definition

MCS 471 Lecture 10
Numerical Analysis
Jan Verschelde, 14 September 2022

Row Pivoting for Numerical Stability

a catastrophic calculation

$$\left\{ \begin{array}{rcl} \epsilon x_1 + a_{1,2} x_2 & = & \epsilon + a_{1,2} \\ a_{2,1} x_1 + a_{2,2} x_2 & = & a_{2,1} + a_{2,2} \end{array} \right. \quad \text{solution} : (x_1, x_2) = (1, 1).$$

where

- $\epsilon > 0$ is a very small number; and
- $a_{1,2}$, $a_{2,1}$, and $a_{2,2}$ are random numbers
  so the determinant is nonzero: $\epsilon a_{2,2} - a_{2,1} a_{1,2} \neq 0$.

Row reduction:

$$A = \left[\begin{array}{cc} \epsilon & a_{1,2} \\ a_{2,1} & a_{2,2} \end{array} \right] \xrightarrow{R_2 := R_2 - \frac{a_{2,1}}{\epsilon} R_1} \left[\begin{array}{cc} \epsilon & a_{1,2} \\ 0 & a_{2,2} - \frac{a_{1,1} a_{2,1}}{\epsilon} \end{array} \right]$$

Then the L and U in the LU factorization are

$$L = \left[\begin{array}{cc} 1 & 0 \\ \frac{a_{2,1}}{\epsilon} & 1 \end{array} \right] \quad \text{and} \quad U = \left[\begin{array}{cc} \epsilon & a_{1,2} \\ 0 & a_{2,2} - \frac{a_{1,2} a_{2,1}}{\epsilon} \end{array} \right].$$

forward and backward substitution

1. Forward substitution to solve $L\mathbf{y} = \mathbf{b}$, $L = \left[\begin{array}{cc} 1 & 0 \\ \frac{a_{2,1}}{\epsilon} & 1 \end{array} \right]$

$$\left\{ \begin{array}{rcl} y_1 & = & \epsilon + a_{1,2} \\ \frac{a_{2,1}}{\epsilon} \; y_1 + y_2 & = & a_{2,1} + a_{2,2} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} y_1 = \epsilon + a_{1,2} \\ y_2 = a_{2,1} + a_{1,2} - \frac{a_{2,1}}{\epsilon}(\epsilon + a_{1,2}) \end{array} \right.$$

   For a tiny $\epsilon > 0$, $y_2 \approx 1/\epsilon$ is huge.

2. Backward substitution to solve $U\mathbf{x} = \mathbf{y}$, $U = \left[\begin{array}{cc} \epsilon & a_{1,2} \\ 0 & a_{2,2} - \frac{a_{1,2}a_{2,1}}{\epsilon} \end{array} \right]$

$$\left\{ \begin{array}{rcrcl} \epsilon x_1 & + & a_{1,2}x_2 & = & \epsilon + a_{1,2} \\ & & \left(a_{2,2} - \frac{a_{2,1}a_{2,2}}{\epsilon} \right) x_2 & = & a_{2,1} + a_{1,2} - \frac{a_{2,1}}{\epsilon}(\epsilon + a_{1,2}) \end{array} \right.$$

   For tiny $\epsilon > 0$: $\dfrac{1}{\epsilon} + 1 = \dfrac{1}{\epsilon}$, so $\dfrac{1}{\epsilon} x_2 = \dfrac{1}{\epsilon}$ implies $x_1 = 0 \neq 1$.

swap rows with a permutation matrix

$$\underbrace{\left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right]}_{P}\underbrace{\left[\begin{array}{cc} \epsilon & a_{1,2} \\ a_{2,1} & a_{2,2} \end{array}\right]}_{A} = \underbrace{\left[\begin{array}{cc} a_{2,1} & a_{2,2} \\ \epsilon & a_{1,2} \end{array}\right]}_{P \star A}$$

The matrix P is a permutation matrix, its inverse $P^{-1} = P$.

$$\begin{array}{rcl} A\mathbf{x} = \mathbf{b} & \Leftrightarrow & P \star A\mathbf{x} = P\mathbf{b} \\ & \Leftrightarrow & LU\mathbf{x} = P\mathbf{b} \end{array}$$

Row reduction:

$$A = \left[\begin{array}{cc} a_{2,1} & a_{2,2} \\ \epsilon & a_{1,2} \end{array}\right] \xrightarrow{R_2 := R_2 - \frac{\epsilon}{a_{2,1}} R_1} \left[\begin{array}{cc} a_{2,1} & a_{2,2} \\ 0 & a_{1,2} - \frac{\epsilon}{a_{2,1}} a_{2,2} \end{array}\right]$$

forward and backward substitution

1. Forward substitution to solve $L\mathbf{y} = P\mathbf{b}$, $L = \begin{bmatrix} 1 & 0 \\ \frac{\epsilon}{a_{2,1}} & 1 \end{bmatrix}$

$$\left\{ \begin{array}{rcl} y_1 & = & a_{2,1} + a_{2,2} \\ \frac{\epsilon}{a_{2,1}} y_1 + y_2 & = & \epsilon + a_{1,2} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} y_1 = a_{2,1} + a_{2,2} \\ y_2 = \epsilon + a_{1,2} - \frac{\epsilon}{a_{2,1}}(a_{2,1} + a_{2,2}) \end{array} \right.$$

2. Backward substitution on $U\mathbf{x} = \mathbf{y}$, $U = \begin{bmatrix} a_{2,1} & a_{2,2} \\ 0 & a_{1,2} - \frac{\epsilon}{a_{2,1}} a_{2,2} \end{bmatrix}$

$$\left\{ \begin{array}{rcl} a_{2,1} x_1 + a_{2,2} x_2 & = & a_{2,1} + a_{2,2} \\ \left(a_{1,2} - \frac{\epsilon}{a_{2,1}} a_{2,2} \right) x_2 & = & \epsilon + a_{1,2} - \frac{\epsilon}{a_{2,1}}(a_{2,1} + a_{2,2}) \end{array} \right.$$

For tiny $\epsilon > 0$, ignore terms with ϵ: $a_{1,2} x_2 \approx a_{1,2}$,
so $x_2 = 1 + O(\epsilon)$ and $x_1 = 1 + O(\epsilon)$.

Row Pivoting for Numerical Stability

1. Motivation
   - the need for pivoting
   - a numerical example
2. PLU Factorization
   - partial pivoting
   - a numerical example
3. Numerical Stability
   - a definition

a numerical example

Exercise 1: Consider the matrix

$$A = \begin{bmatrix} 10^{-8} & 1 \\ 1 & 1 \end{bmatrix}$$

and let $\mathbf{b} = A \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. So the solution to $A\mathbf{x} = \mathbf{b}$ is $(x_1 = 1, x_2 = 1)$.

1. Compute the LU factorization of A.
2. Use the LU factorization to solve $A\mathbf{x} = \mathbf{b}$.

What is the error on the solution *without* and *with* pivoting?
Show all your work, either with a Julia session or by hand.

matrices with small diagonal elements

Exercise 2: Generate random matrices A of dimension n, for n from 3 to 10, replacing of each A the diagonal D by $D10^{-k}$, $R = A - D + D10^{-k}$, for k ranging from 1 to 10.

```
using LinearAlgebra; A = rand(n, n); D = Diagonal(A)
R = A - D + D*10^(-k)
```

Let $\mathbf{x}$ be an n-dimensional vector of ones and $\mathbf{b} = R\mathbf{x}$, so the system $R\mathbf{x} = \mathbf{b}$ has all ones as its exact solution.

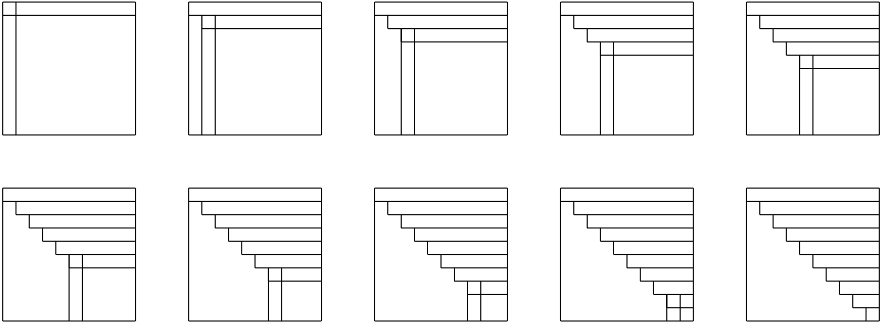
For n from 3 to 10 and k from 1 to 10, compute L and U of the LU factorization without pivoting of the matrices R with the `lufac` function of lecture 8, followed by the forward substitution $\mathbf{y} = L\backslash\mathbf{b}$ and $\bar{\mathbf{x}} = U\backslash\mathbf{y}$.

Make a table of errors $\|\bar{\mathbf{x}} - \mathbf{x}\|$, the norm of the difference between the computed and the exact solution. When printing the table of errors, format the numbers so only one decimal after the point appears.

How do the errors change as a function of n and k?

Row Pivoting for Numerical Stability

LU factorization with partial pivoting



for column $j = 1, 2, \ldots, n-1$ in A do

1. find the largest element $|a_{i,j}|$ in column j (for $i \geq j$);
2. if $i \neq j$, then swap rows i and j;
3. for $i = j+1, \ldots n$, for $k = j+1, \ldots, n$ do $a_{i,k} := a_{i,k} - \left(\dfrac{a_{i,j}}{a_{j,j}} \right) a_{j,k}$.

the LU factorization in Julia

```
julia> using LinearAlgebra

julia> A = rand(10,10);

julia> L, U, P = lu(A);

julia> norm(A[P, :] - L*U)
3.5961318749671356e-16

julia> transpose(P)
1x10 RowVector{Int64,Array{Int64,1}}:
 4  9  3  10  6  8  7  1  2  5
```

`P` stores the pivoting information

`A[P, :]` selects the rows of `A` as defined by `P`

permutation matrices

Definition

A permutation matrix has exactly one 1 in every row and every column and all its other elements are 0.

For example, for `P = 4 9 3 10 6 8 7 1 2 5` we have:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} x_4 \\ x_9 \\ x_3 \\ x_{10} \\ x_6 \\ x_8 \\ x_7 \\ x_1 \\ x_2 \\ x_5 \end{bmatrix}$$

the LU factorization algorithm with row pivoting

Given is an n-by-n matrix A:

$p := [1, 2, \ldots, n]$
for $j = 1, 2, \ldots, n$ do
  let k be such that $|a_{k,j}| = \max\limits_{i=j}^{n} |a_{i,j}|$
  if $k \neq j$ then
    $A[k,:], A[j,:] := A[j,:], A[k,:]$, (swap row j with row k)
    $p[j], p[k] := p[k], p[j]$ (swap $p[j]$ with $p[k]$)
  for $i = j+1, \ldots, n$ do
    $a_{i,j} := a_{i,j}/a_{j,j}$ ($a_{i,j}$ is multiplier)
    for $k = j+1, j+2, \ldots, n$ do
      $a_{i,k} := a_{i,k} - a_{i,j}a_{j,k}$ (reduce row i)

The factor L is on the lower diagonal part of A
and the factor U is on the diagonal and upper diagonal part of A.
The pivots in p define the permutation matrix P, so $P \star A = L \star U$.

swapping rows of a matrix in Julia

Swapping rows can be done with one statement:

```
julia> A = rand(3,3)
3×3 Array{Float64,2}:
 0.544985  0.475856  0.339745
 0.632158  0.346131  0.504592
 0.716925  0.340208  0.971623

julia> A[1,:], A[3,:] = A[3,:], A[1,:]

julia> A
3×3 Array{Float64,2}:
 0.716925  0.340208  0.971623
 0.632158  0.346131  0.504592
 0.544985  0.475856  0.339745

julia>
```

Row Pivoting for Numerical Stability

1. Motivation
   - the need for pivoting
   - a numerical example
2. PLU Factorization
   - partial pivoting
   - a numerical example
3. Numerical Stability
   - a definition

a numerical example

Consider

```
         [  9.229E-02  -1.324E+00   1.976E+00  ]
     A = [ -6.501E-01   1.201E+00  -3.308E-01  ].
         [  2.245E+00  -1.265E+00  -1.277E+00  ]
```

Compute the LU decomposition of `A` with partial pivoting.
Use 4 decimal places with rounding,
and write all floating-point numbers in scientific format.

```
          [  2.245E+00  -1.265E+00  -1.277E+00  ] 3
A ----> [ -6.501E-01   1.201E+00  -3.308E-01  ] 2
          [  9.229E-02  -1.324E+00   1.976E+00  ] 1
```

the numerical example continued

```
          [  2.245E+00  -1.265E+00  -1.277E+00  ] 3
 A ----> [ -6.501E-01   1.201E+00  -3.308E-01  ] 2
          [  9.229E-02  -1.324E+00   1.976E+00  ] 1

          -.6501     [                                    ]
R2 := R2 - ------ R1  [  2.245E+00  -1.265E+00  -1.277E+00  ] 3
           2.245     [                                    ]
---------------------> [ -2.896E-01   8.347E-01  -7.006E-01  ] 2
          0.09229    [                                    ]
R3 := R3 - ------- R1 [  4.111E-02  -1.272E+00   2.028E+00  ] 1
           2.245     [                                    ]

                     [  2.245E+00  -1.265E+00  -1.277E+00  ] 3
---------------------> [  4.111E-02  -1.272E+00   2.028E+00  ] 1
                     [ -2.896E-01   8.347E-01  -7.006E-01  ] 2
```

the numerical example continued

```
                       [  2.245E+00  -1.265E+00  -1.277E+00  ] 3
---------------------> [  4.111E-02  -1.272E+00   2.028E+00  ] 1
                       [ -2.896E-01   8.347E-01  -7.006E-01  ] 2

            .8347      [                                     ]
R3 := R3 - ------ R2   [  2.245E+00  -1.265E+00  -1.277E+00  ] 3
           -1.272      [                                     ]
---------------------> [  4.111E-02  -1.272E+00   2.028E+00  ] 1
                       [                                     ]
                       [ -2.896E-01  -6.562E-01   6.302E-01  ] 2
                       [                                     ]
```

the numerical example completed

```
[  2.245E+00  -1.265E+00  -1.277E+00  ] 3
[  4.111E-02  -1.272E+00   2.028E+00  ] 1
[ -2.896E-01  -6.562E-01   6.302E-01  ] 2

    [ 0 0 1 ]       [      1            0             0      ]
P = [ 1 0 0 ]  L =  [  4.111E-02        1             0      ]
    [ 0 1 0 ]       [ -2.896E-01  -6.562E-01          1      ]

                    [  2.245E+00  -1.265E+00  -1.277E+00  ]
               U =  [      0      -1.272E+00   2.028E+00  ]
                    [      0           0       6.302E-01  ]
```

two exercises

Exercise 3:

Consider the matrix $A = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}$.

Apply row pivoting to compute the LU factorization of A.
Work in exact rational arithmetic.
Write the matrices P, L, and U. Verify that $PA = L \star U$.

Exercise 4:

Consider (again) the matrix $A = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}$.

Compute the LU decomposition of `A` with row pivoting.
Use 4 decimal places with rounding,
and write all floating-point numbers in scientific format.
Compare the results with the results of Exercise 3.

the Julia function `lufac`

```
"""
    lufac(mat::Array{Float64,2}, verbose::Bool=true)

returns the matrices L, U, and P in an LU factorization
of mat, with row pivoting.
If verbose, writes in every step the matrix.

Example:

    A = rand(4,4)
    L, U, P = lufac(A)
"""
function lufac(mat::Array{Float64,2},
               verbose::Bool=true)
```

running the posted program `plufac.jl`

```
A matrix :
3×3 Array{Float64,2}:
  9.229e-02  -1.324e+00   1.976e+00
 -6.501e-01   1.201e+00  -3.308e-01
  2.245e+00  -1.265e+00  -1.277e+00
Step 1, the pivot row : 3
The matrix after swap :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
 -6.501e-01   1.201e+00  -3.308e-01
  9.229e-02  -1.324e+00   1.976e+00
The matrix after the update :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
 -2.896e-01   8.347e-01  -7.006e-01
  4.111e-02  -1.272e+00   2.028e+00
```

the output continues ...

```
Step 2, the pivot row : 3
The matrix after swap :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
  4.111e-02  -1.272e+00   2.028e+00
 -2.896e-01   8.347e-01  -7.006e-01
The matrix after the update :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
  4.111e-02  -1.272e+00   2.028e+00
 -2.896e-01  -6.562e-01   6.305e-01
Step 3, the pivot row : 3
The matrix after the update :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
  4.111e-02  -1.272e+00   2.028e+00
 -2.896e-01  -6.562e-01   6.305e-01
```

the output continues ...

```
The output of the LU factorization :
The pivots : [3, 1, 2]
The lower triangular matrix :
3×3 Array{Float64,2}:
  1.000e+00   0.000e+00  0.000e+00
  4.111e-02   1.000e+00  0.000e+00
 -2.896e-01  -6.562e-01  1.000e+00
The upper triangular matrix :
3×3 Array{Float64,2}:
 2.245e+00  -1.265e+00  -1.277e+00
 0.000e+00  -1.272e+00   2.028e+00
 0.000e+00   0.000e+00   6.305e-01
```

the output continues ...

```
Verification of the output :
The product of lower with upper :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
  9.229e-02  -1.324e+00   1.976e+00
 -6.501e-01   1.201e+00  -3.308e-01
The permuted input matrix :
3×3 Array{Float64,2}:
  2.245e+00  -1.265e+00  -1.277e+00
  9.229e-02  -1.324e+00   1.976e+00
 -6.501e-01   1.201e+00  -3.308e-01
The difference : 2.220e-16
```

again matrices with small diagonal elements

Exercise 5: Generate random matrices A of dimension n, for n from 3 to 10, replacing of each A the diagonal D by $D10^{-k}$, $R = A - D + D10^{-k}$, for k ranging from 1 to 10.

```
using LinearAlgebra; A = rand(n, n); D = Diagonal(A)
R = A - D + D*10^(-k)
```

Let $\mathbf{x}$ be an n-dimensional vector of ones and $\mathbf{b} = R\mathbf{x}$, so the system $R\mathbf{x} = \mathbf{b}$ has all ones as its exact solution.

For n from 3 to 10 and k from 1 to 10, do

```
L, U, P = lu(R); y = L\b[P]; x = U\y
```

Make a table of errors $\|\overline{\mathbf{x}} - \mathbf{x}\|$, the norm of the difference between the computed and the exact solution. When printing the table of errors, format the numbers so only one decimal after the point appears.

How do the errors change as a function of n and k?

Row Pivoting for Numerical Stability

1. Motivation
   - the need for pivoting
   - a numerical example
2. PLU Factorization
   - partial pivoting
   - a numerical example
3. Numerical Stability
   - a definition

conditioning and numerical stability

The random matrices considered in exercises 2 and 5
have small elements on their diagonal
but they are *not* ill-conditioned.

Row reduction without pivoting is an example of
a numerically instable algorithm.

Row reduction with pivoting is numerically stable.

Definition

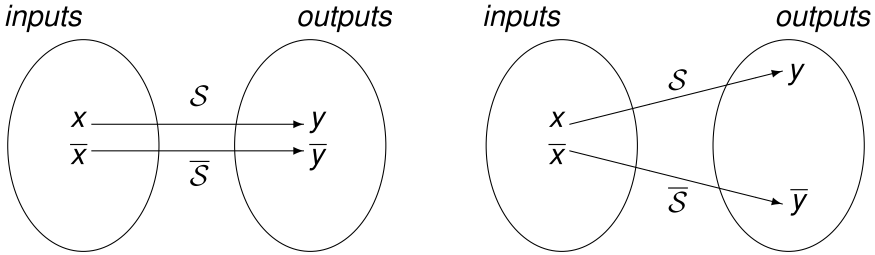
An algorithm to compute $y = f(x)$ is *numerically stable*
if for any x and small Δx: we have

$$\overline{y} = f(x + \Delta x) \quad \text{and} \quad \|\overline{y} - y\| \approx \|\Delta x\|.$$

If the errors in the input are small,
then the output of a numerically stable algorithm has also small errors.

numerical stability illustrated

Let $y = \mathcal{S}(x)$ be exact, while $\overline{y} = \overline{\mathcal{S}}(x)$ is computed numerically.



$\overline{\mathcal{S}}$ is numerically stable

$\overline{\mathcal{S}}$ is *not* numerically stable

Definition

An algorithm to compute $y = f(x)$ is *numerically stable* if for any x and small Δx: we have

$$\overline{y} = f(x + \Delta x) \quad \text{and} \quad \|\overline{y} - y\| \approx \|\Delta x\|.$$