Numerical Methods of Linear Algebra for Sparse Matrices

Lecture 4

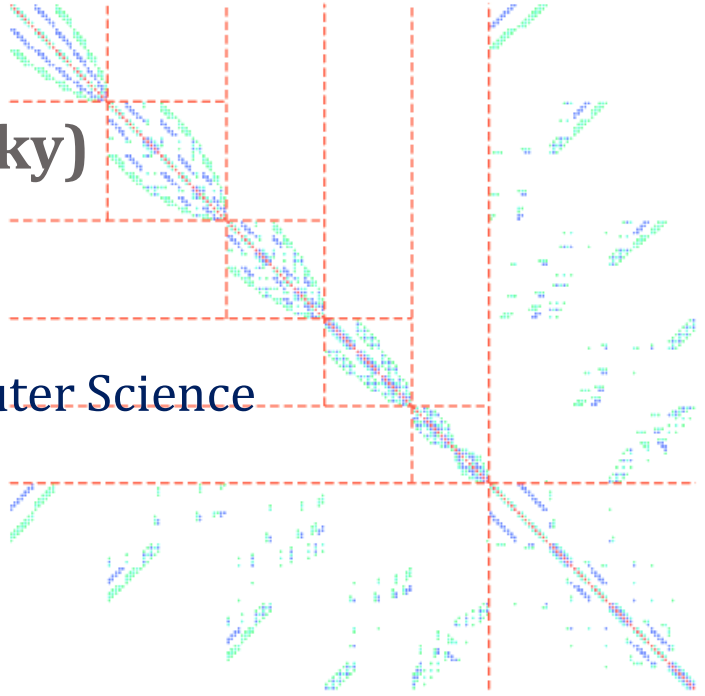
Matrix factorizations (QR, LU, Cholesky)

Anna Nasedkina, PhD, Assoc. prof.

Department of Mathematical Modeling

Institute of Mathematics, Mechanics and Computer Science

Southern Federal University



Outline

- QR-factorization
- LU-factorization
- Cholesky factorization

QR-factorization

- Factorization of a matrix is its representation as a product of several matrices, usually two or three
- QR-factorization of a matrix $A \in \mathbb{C}^{n\times m}$, $n \geq m$

$A = QR$, where Q is unitary and R is upper triangular

The sizes of these two matrices can be diffent:

- Thin (reduced) QR-factorization is

$A = QR$, where $Q \in \mathbb{C}^{n\times m}$ and $R \in \mathbb{C}^{m\times m}$

- (Full) QR-factorization is

$A = QR$, where $Q \in \mathbb{C}^{n\times n}$ and $R \in \mathbb{C}^{n\times m}$

Consider matrix A column-wise:

$$A = \begin{pmatrix} | & | & & | \\ a_1 & a_2 & \dots & a_m \\ | & | & & | \end{pmatrix}, a_j \in \mathbb{C}^n, j = \overline{1,m}$$

- G-S process applied to the columns of A gives a thin QR-factorization
- Also, QR-factorization can be computed by Givens rotations or Housholder reflections

Theorems on QR-factorization: existence and uniqueness

- It's possible to find QR-factorization for any matrix
- In general, QR-factorization is not unique

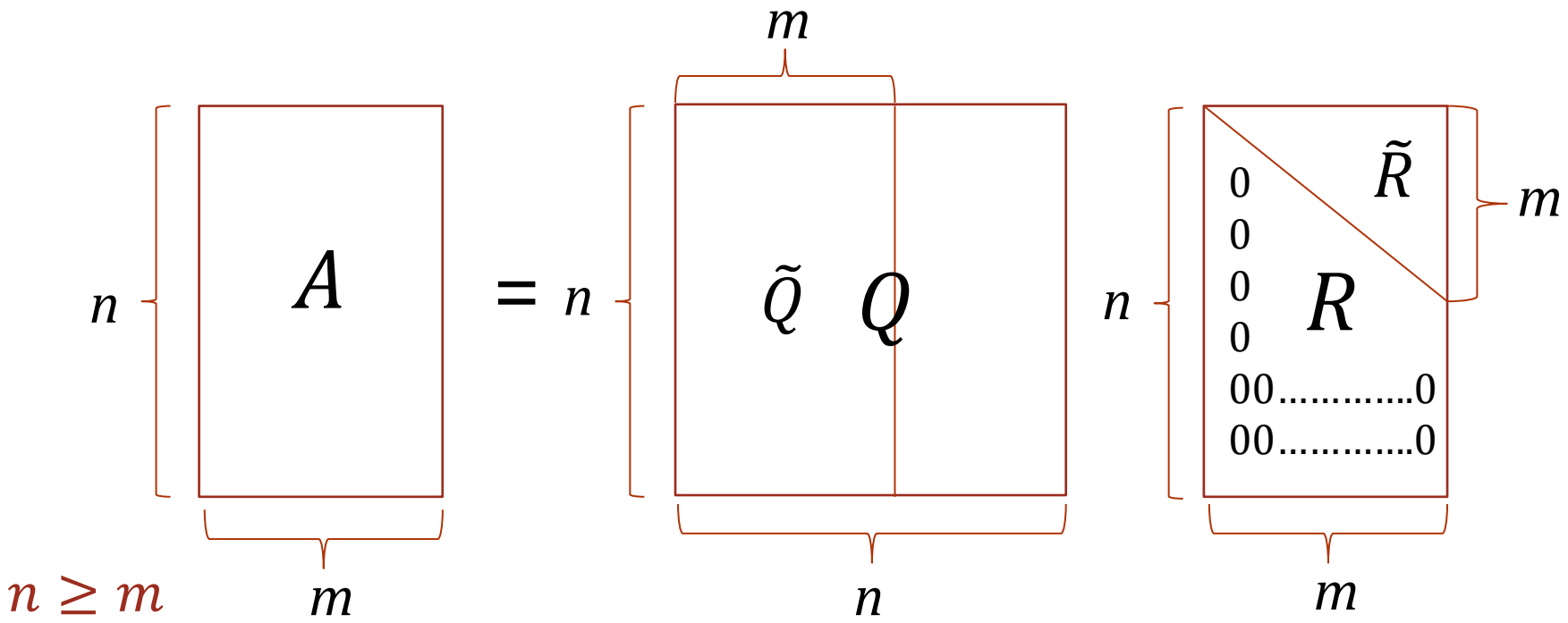
Theorem 1. $\forall A \in \mathbb{C}^{n\times m}, n \geq m$ $\exists$ unitary matrix $Q \in \mathbb{C}^{n\times n}$ and upper triangular matrix $R \in \mathbb{C}^{n\times m}$, such that $A = QR$.

In addition, $A = \tilde{Q}\tilde{R}$, where $\tilde{Q} \in \mathbb{C}^{n\times m}$ is Q with the last $n-m$ columns skipped and $\tilde{R} \in \mathbb{C}^{m\times m}$ is R with the last $n-m$ zero rows skipped

Theorem 2. For any full rank matrix $A \in \mathbb{C}^{n\times m}, n \geq m,\ rank(A) = m$ thin QR-factorization is unique ($\exists!\ \tilde{Q}$ and $\exists!\ \tilde{R}$), when all diagonal entries of R are positive: $r_{ii} > 0$

In the case when diagonal entries are not restricted to be all positive, full QR-factorization is not unique: $A = Q_1 R_1 = Q_2 R_2$

Relation between thin and full QR



- $\tilde{Q}$ is Q with the last $n - m$ column skipped, $\tilde{R}$ is R with the last $n - m$ rows skipped

QR

full

$$A_{n \times m} = Q_{n \times n} * R_{n \times m}$$

thin

$$A_{n \times m} = \tilde{Q}_{n \times m} * \tilde{R}_{m \times m}$$

QR-factorization in Matlab

QR-factorization for rectangular matrix n>m

```
n=5;m=3;
A=rand(n,m)


[Qf,Rf]=qr(A) % full qr




[Qt,Rt]=qr(A,0) % thin qr




Qf*Rf % gives A
```

```
A = 5×3
    0.5285    0.6892    0.9133
    0.1656    0.7482    0.1524
    0.6020    0.4505    0.8258
    0.2630    0.0838    0.5383
    0.6541    0.2290    0.9961

Qf = 5×5
   -0.4894   -0.3591    0.5737   -0.5021   -0.2240
   -0.1534   -0.8278   -0.2394    0.2979    0.3810
   -0.5575    0.0406   -0.4975    0.2243   -0.6243
   -0.2435    0.1696    0.5783    0.7589    0.0403
   -0.6057    0.3943   -0.1776   -0.1813    0.6429

Rf = 5×3
   -1.0799   -0.8623   -1.6652
         0   -0.7440    0.0635
         0         0    0.2111
         0         0         0
         0         0         0

Qt = 5×3
   -0.4894   -0.3591    0.5737
   -0.1534   -0.8278   -0.2394
   -0.5575    0.0406   -0.4975
   -0.2435    0.1696    0.5783
   -0.6057    0.3943   -0.1776

Rt = 3×3
   -1.0799   -0.8623   -1.6652
         0   -0.7440    0.0635
         0         0    0.2111

ans = 5×3
    0.5285    0.6892    0.9133
    0.1656    0.7482    0.1524
    0.6020    0.4505    0.8258
    0.2630    0.0838    0.5383
    0.6541    0.2290    0.9961
```

```
isequal(A,Qf*Rf)
norm(A-Qf*Rf)
norm(A-Qt*Rt)
```

```
ans = logical
   0

ans = 3.6653e-16

ans = 3.6653e-16
```

QR-factorization in Matlab

QR-factorization for square matrix n=m

```
n=4;
A=rand(n)



[Qf,Rf]=qr(A) % full qr









[Qt,Rt]=qr(A,0) % thin qr
```

```
A = 4x4
    0.1419    0.9595    0.9340    0.3922
    0.4218    0.6557    0.6787    0.6555
    0.9157    0.0357    0.7577    0.1712
    0.7922    0.8491    0.7431    0.7060

Qf = 4x4
   -0.1100   -0.7517    0.5635    0.3246
   -0.3269   -0.3239    0.0155   -0.8877
   -0.7099    0.5041    0.4844    0.0860
   -0.6141   -0.2756   -0.6691    0.3150

Rf = 4x4
   -1.2900   -0.8667   -1.3189   -0.8125
         0   -1.1497   -0.7448   -0.6154
         0         0    0.4066   -0.1583
         0         0         0   -0.2174

Qt = 4x4
   -0.1100   -0.7517    0.5635    0.3246
   -0.3269   -0.3239    0.0155   -0.8877
   -0.7099    0.5041    0.4844    0.0860
   -0.6141   -0.2756   -0.6691    0.3150

Rt = 4x4
   -1.2900   -0.8667   -1.3189   -0.8125
         0   -1.1497   -0.7448   -0.6154
         0         0    0.4066   -0.1583
         0         0         0   -0.2174
```

QR-factorization in Matlab

QR-factorization for rectangular matrix n<m

```
n=3;m=4;
A=rand(n,m)


[Qf,Rf]=qr(A) % full qr




[Qt,Rt]=qr(A,0) % thin qr
```

```
A = 3x4
    0.0318    0.0971    0.3171    0.4387
    0.2769    0.8235    0.9502    0.3816
    0.0462    0.6948    0.0344    0.7655

Qf = 3x3
   -0.1127   -0.0142   -0.9935
   -0.9801   -0.1629    0.1135
   -0.1634    0.9865    0.0044

Rf = 3x4
   -0.2825   -0.9316   -0.9727   -0.5485
         0    0.5500   -0.1253    0.6868
         0         0   -0.2071   -0.3892

Qt = 3x3
   -0.1127   -0.0142   -0.9935
   -0.9801   -0.1629    0.1135
   -0.1634    0.9865    0.0044

Rt = 3x4
   -0.2825   -0.9316   -0.9727   -0.5485
         0    0.5500   -0.1253    0.6868
         0         0   -0.2071   -0.3892
```

QR-factorization in Matlab

QR-factorization for rectangular matrix n<m

```
n=3;m=4;
A=rand(n,m)


[Qf,Rf]=qr(A) % full qr




[Qt,Rt]=qr(A,0) % thin qr
```

```
A = 3x4
    0.0318    0.0971    0.3171    0.4387
    0.2769    0.8235    0.9502    0.3816
    0.0462    0.6948    0.0344    0.7655

Qf = 3x3
   -0.1127   -0.0142   -0.9935
   -0.9801   -0.1629    0.1135
   -0.1634    0.9865    0.0044

Rf = 3x4
   -0.2825   -0.9316   -0.9727   -0.5485
         0    0.5500   -0.1253    0.6868
         0         0   -0.2071   -0.3892

Qt = 3x3
   -0.1127   -0.0142   -0.9935
   -0.9801   -0.1629    0.1135
   -0.1634    0.9865    0.0044

Rt = 3x4
   -0.2825   -0.9316   -0.9727   -0.5485
         0    0.5500   -0.1253    0.6868
         0         0   -0.2071   -0.3892
```

Using QR-factorization to solve linear system

Any matrix factorization can be used to either solve or analyze some problem.

Use QR-factorization to solve the system $Ax = b$

If $Ax = b \Rightarrow x = A^{-1}b$ (this is formal writing, as we shall never compute the inverse A^{-1} in numerical algorithms).

If $A = QR \Rightarrow x = (QR)^{-1}b = R^{-1}Q^{-1}b = R^{-1}Q^{H}b$

($Q^{-1} = Q^{H}$, as Q is unitary).

Hence $x = R^{-1}Q^{H}b$ is the solution of $Rx = Q^{H}b$

R is upper triangular matrix, so this system is faster to solve.

Matlab:

- [Q,R]=qr(A)
- x=R\(Q'*b)

Matrix left division **mldivide, \: A\B ~ inv(A)*B**

LU- and PLU-factorization

Consider a nonsingular square matrix $A \in \mathbb{C}^{n\times n}$

- LU-factorization of a matrix $A \in \mathbb{C}^{n\times n}$

$A = LU$, where L is lower triangular and U is upper triangular

- LU-factorization with partial pivoting of a matrix $A \in \mathbb{C}^{n\times n}$

$PA = LU$, where P is permutation matrix

- Permutation matrix P is the identity matrix with permuted columns:

If $I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix} = \begin{pmatrix} | & | & & | \\ e_1 & e_2 & \dots & e_n \\ | & | & & | \end{pmatrix}$, where e_j is unit ort (it contains 1 at j-th position and all zeros as other elements), $j = \overline{1,n}$; then P is permutation of $\{e_j\}$, i.e. columns of P are $\{e_j\}$, taken in a certain order.

Property. Permutation matrix is orthogonal: $P^{-1} = P^T$

- PA permutes the rows of A, AP permutes the columns of A

Theorems on LU-factorization: existence and uniqueness

- It's not always possible to find LU-factorization for any matrix
- However, PLU-factorization is possible to find for any matrix
- In general, LU-factorization is not unique

Theorem 1. $\forall$ nonsingular $A \in \mathbb{C}^{n\times n}$ $\exists\, P, L, U \in \mathbb{C}^{n\times n}$ such that $A = P^{-1}LU$, where P is permutation matrix, L is lower triangular and U is upper triangular matrix.

Theorem 2. Assume nonsingular $A \in \mathbb{C}^{n\times n}$ has LU-factorization: $A = LU$. Then for any set of nonzero numbers $\{c_i\}$, $c_i \neq 0$, $i = \overline{1,n}$ there is another LU-factorization: $A = \tilde{L}\tilde{U}$, where $\{c_i\}$ are diagonal entries of either $\tilde{L}$ or $\tilde{U}$.

Theorem 3. $\forall$ nonsingular $A \in \mathbb{C}^{n\times n}$ the factorization $A = LU$, where $l_{11}=l_{22}=\ldots=l_{nn}=1$ is unique (if exists).

Note of uniqueness of LU-factorization

If exists, LU-factorization $\forall$ nonsingular $A \in \mathbb{C}^{n\times n}$ is defined up to at least n degrees of freedom, because if $A = LU \Rightarrow A = \tilde{L}\tilde{U},$ where $\tilde{L} = LD,\ \tilde{U} = D^{-1}U$ or $\tilde{L} = LD^{-1},\ \tilde{U} = DU,$ where D is any diagonal matrix.

- In Matlab we can compute LD^{-1} as **L/D** (right division or **mrdivide(L,D)**
- **Matrix right division B/A = B*inv(A)** solves the system of linear equations X*A = B for X

LU-factorization in Matlab

```
n=3;
A=rand(n)
[L,U]=lu(A) % computes upper triangular matrix U
% and a permuted lower triangular matrix L such that
% A = L*U.









L*U % gives A


isequal(A,L*U) % sometimes gives logical 1

norm(A-L*U)

[L1,U1,P]=lu(A) % computes permutation matrix P,
% unit lower triangular L1 and upper triangular U1,
% such that A = P'*L1*U1.
```

```
A = 3×3
    0.1890    0.3685    0.0811
    0.6868    0.6256    0.9294
    0.1835    0.7802    0.7757

L = 3×3
    0.2751    0.3203    1.0000
    1.0000         0         0
    0.2672    1.0000         0

U = 3×3
    0.6868    0.6256    0.9294
         0    0.6131    0.5274
         0         0   -0.3435

ans = 3×3
    0.1890    0.3685    0.0811
    0.6868    0.6256    0.9294
    0.1835    0.7802    0.7757


ans = logical
   0

ans = 2.7756e-17
L1 = 3×3
    1.0000         0         0
    0.2672    1.0000         0
    0.2751    0.3203    1.0000

U1 = 3×3
    0.6868    0.6256    0.9294
         0    0.6131    0.5274
         0         0   -0.3435

P = 3×3
     0     1     0
     0     0     1
     1     0     0
```

LU-factorization in Matlab

```
% A = P'*L1*U1
A-P'*L1*U1
norm(A-P'*L1*U1)

% L1 is P*L

PL=P*L

L1=L1
L1-P*L
```

```
ans = 3×3
10^-16 ×
         0         0         0
         0         0         0
   -0.2776         0         0

ans = 2.7756e-17
PL = 3×3
    1.0000         0         0
    0.2672    1.0000         0
    0.2751    0.3203    1.0000

L1 = 3×3
    1.0000         0         0
    0.2672    1.0000         0
    0.2751    0.3203    1.0000

ans = 3×3
     0     0     0
     0     0     0
     0     0     0
```

LU-factorization in Matlab: illustration of non-uniqueness

```
A=rand(3);
A=A*A'
```

```
A = 3×3
    1.3187    0.6497    1.0662
    0.6497    0.9577    0.2740
    1.0662    0.2740    0.9718
```

```
[l,u]=lu(A) %generated lu-factorization
% with strictly lower-triangular l
```

```
l = 3×3
    1.0000         0         0
    0.4927    1.0000         0
    0.8085   -0.3941    1.0000

u = 3×3
    1.3187    0.6497    1.0662
         0    0.6377   -0.2513
         0         0    0.0107
```

```
A-l*u
```

```
ans = 3×3
10^-15 ×
         0         0         0
         0         0         0
         0         0    0.1110
```

```
D=diag(rand(1,3))
```

```
D = 3×3
    0.4624         0         0
         0    0.4243         0
         0         0    0.4609
```

```
L1=l*D
```

```
L1 = 3×3
    0.4624         0         0
    0.2278    0.4243         0
    0.3739   -0.1672    0.4609
```

```
U1=inv(D)*u
```

```
U1 = 3×3
    2.8515    1.4049    2.3055
         0    1.5027   -0.5923
         0         0    0.0233
```

```
A-L1*U1
```

```
ans = 3×3
10^-15 ×
   -0.2220         0         0
   -0.1110         0         0
   -0.2220         0    0.1110
```

LU-factorization in Matlab: illustration of non-uniqueness

```
L2=l*inv(D)
```

```
L2 = 3×3
    2.1624         0         0
    1.0654    2.3566         0
    1.7484   -0.9288    2.1696
```

```
L2=l/D %alternative way to compute using right-division
```

```
L2 = 3×3
    2.1624         0         0
    1.0654    2.3566         0
    1.7484   -0.9288    2.1696
```

```
U2=D*u
```

```
U2 = 3×3
    0.6098    0.3004    0.4931
         0    0.2706   -0.1066
         0         0    0.0049
```

```
A-L2*U2
```

```
ans = 3×3
10^-15 ×
   -0.2220   -0.1110         0
         0   -0.1110         0
         0         0    0.1110
```

Using LU-factorization to solve linear system

Use LU-factorization to solve the system $Ax = b$

If $A = LU \Rightarrow LUx = b.$

Denote $Ux = y \Rightarrow Ly = b$. We get two systems:

1) System $Ly = b$ with lower-triangular matrix

(can be solved using forward substitution)

$Ly = b \Rightarrow$ formal solution $y = L^{-1}b$

2) System $Ux = y$ with upper-triangular matrix

(can be solved using back substitution)

$Ux = y \Rightarrow$ formal solution $x = U^{-1}y$

Hence $x = U^{-1}L^{-1}b$ is the solution of initial system $Ax = b$

Matlab:

- [L,U]=lu(A)
- x=U\(L\b)

Cholesky factorization, theorems on existence and uniqueness

When $A = A^H$, we can try to compute U as L^H to preserve symmetry:

$A = LL^H$ or $A = U^H U = R^H R$

(R is usual notation for upper triangular matrix in Cholesky factorization)

- Cholesky-factorization of a Hermitian matrix $A \in \mathbb{C}^{n\times n}$

$A = LL^H$, where L is lower triangular

- Cholesky factorization for Hermitian matrix is not always possible, the matrix should be also *positive definite* (definition will be given later)

Theorem 1. If for nonsingular $A \in \mathbb{C}^{n\times n}$ $\exists\, L \in \mathbb{C}^{n\times n}$ such that $A = LL^H$, then L is unique.

Theorem 2. $\forall\, A \in \mathbb{C}^{n\times n}$ $\exists\, L \in \mathbb{C}^{n\times n} : A = LL^H \Leftrightarrow$ A is Hermitian positive definite.

Theorem 3. Cholesky-factorization is unique, if all diagonal entries of R are positive.

Cholesky factorization in Matlab

```
n=3;
A=rand(n);
A=A*A'
R=chol(A) % for symmetric positive definite matrix
% computes upper triangular R, such that A=R'*R
A-R'*R
norm(A-R'*R)
```

```
A = 3×3
    0.7209    0.8453    0.6216
    0.8453    1.3159    0.7771
    0.6216    0.7771    0.5943

R = 3×3
    0.8490    0.9956    0.7321
         0    0.5698    0.0846
         0         0    0.2261

ans = 3×3
10^-15 ×
         0   -0.1110         0
   -0.1110         0         0
         0         0         0

ans = 1.1102e-16
```

```
L=chol(A,'lower') % for symmetric positive definite matrix
% computes lower triangular such that A=L*L'
A-L*L'
norm(A-L*L')
```

```
L = 3×3
    0.8490         0         0
    0.9956    0.5698         0
    0.7321    0.0846    0.2261

ans = 3×3
10^-15 ×
         0   -0.1110         0
   -0.1110         0         0
         0         0         0

ans = 1.1102e-16
```

```
[R,flag] = chol(A) % also returns the output flag
% indicating whether A is symmetric positive definite.
% For symmetric positive definite flag = 0,
% otherwise flag is positive.
```

```
R = 3×3
    0.8490    0.9956    0.7321
         0    0.5698    0.0846
         0         0    0.2261

flag = 0
```

Ways to solve linear system in Matlab

Consider linear system Ax=b

1. x=A\b or x=mldivide(A,b)
2. x=linsolve(A,b)
3. x=inv(A)*b
4. Use factorizations of A