In [1]:
```python
import sympy
from sympy import Matrix, S, Symbol, symbols, I, zeros, eye, simplify, expand, expand_complex
import numpy as np
from numpy import random
```

Занятие 18

Алгебра

Матричные разложения: Холецкого, LDL, LU, QR. Жорданова форма.

Разложение Холецкого

$A = L \cdot L^T$ для симметричной вещественной матрицы A

$A = L \cdot L^H$ для положительно определенной эрмитовой матрицы A

L - левая треугольная матрица.

Пример 1.

Построим разложение Холецкого матриц

$$A = \begin{pmatrix} 2 & 3 & 5 \\ 3 & -2 & 1 \\ 5 & 1 & 6 \end{pmatrix} \quad B = \begin{pmatrix} 12 & 3+I & 5 \\ 3-I & 2 & 1-I \\ 5 & 1+I & 6 \end{pmatrix}$$

In [2]:
```python
A = Matrix([[2, 3, 5], [3, -2, 1], [5, 1, 6]])
B = Matrix([[12, 3 + I, 5], [3 - I, 2, 1 - I], [5, 1 + I, 6]])
LA = A.cholesky(hermitian=False)
LB = B.cholesky()
display('LA', LA, 'simplify(LA*LA.T - A)', simplify(LA*LA.T - A),
        'B.is_positive_definite', B.is_positive_definite,
        'LB', LB, 'simplify(LB)', simplify(LB),
        'simplify(expand(LB))', simplify(expand(LB)),
        'simplify(LB*LB.H - B)', simplify(LB*LB.H - B))
```

'LA'

$$\begin{bmatrix} \sqrt{2} & 0 & 0 \\ \frac{3\sqrt{2}}{2} & \frac{\sqrt{26}i}{2} & 0 \\ \frac{5\sqrt{2}}{2} & \frac{\sqrt{26}i}{2} & 0 \end{bmatrix}$$

'simplify(LA*LA.T - A)'

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

'B.is_positive_definite'

True

'LB'

$$\begin{bmatrix} 2\sqrt{3} & 0 & 0 \\ \frac{\sqrt{3}(3-i)}{6} & \sqrt{-\frac{(3-i)(3+i)}{12}+2} & 0 \\ \frac{5\sqrt{3}}{6} & \frac{-\frac{1}{4}+\frac{7i}{12}}{\sqrt{-\frac{(3-i)(3+i)}{12}+2}} & \sqrt{-\frac{\left(-\frac{1}{4}-\frac{7i}{12}\right)\left(-\frac{1}{4}+\frac{7i}{12}\right)}{-\frac{(3-i)(3+i)}{12}+2}+\frac{47}{12}} \end{bmatrix}$$

'simplify(LB)'

$$\begin{bmatrix} 2\sqrt{3} & 0 & 0 \\ \frac{\sqrt{3}(3-i)}{6} & \frac{\sqrt{42}}{6} & 0 \\ \frac{5\sqrt{3}}{6} & \frac{\sqrt{42}(1+i)(2+5i)}{84} & \frac{5\sqrt{7}}{7} \end{bmatrix}$$

'simplify(expand(LB))'

$$\begin{bmatrix} 2\sqrt{3} & 0 & 0 \\ \frac{\sqrt{3}(3-i)}{6} & \frac{\sqrt{42}}{6} & 0 \\ \frac{5\sqrt{3}}{6} & \frac{\sqrt{42}(-3+7i)}{84} & \frac{5\sqrt{7}}{7} \end{bmatrix}$$

'simplify(LB*LB.H - B)'

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

LDL разложение

$A = LDL^T$ для симметричной вещественной матрицы A

$A = LDL^H$ для положительно определенной эрмитовой матрицы A

L - левая треугольная матрица. $A = LDL^H$ для матрицы A, B - бидиагональная матрица.

Пример 2.

Построим LDL разложение для матриц Примера 1

In [3]:
```python
A = Matrix([[2, 3, 5], [3, -2, 1], [5, 1, 6]])
B = Matrix([[12, 3 + I, 5], [3 - I, 2, 1 - I], [5, 1 + I, 6]])
LA, DA = A.LDLdecomposition(hermitian=False)
LB, DB = B.LDLdecomposition()
display('LDLA', LA, DA, 'simplify(LA*DA*LA.T - A)', simplify(LA*DA*LA.T - A),
        'B.is_positive_definite', B.is_positive_definite,
        'LDLB', LB, DB, 'simplify(LB)', simplify(LB),
        'simplify(LB*DB*LB.H - B)', simplify(LB*DB*LB.H - B))
```

'LDLA'

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{5}{2} & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & -\frac{13}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

'simplify(LA*DA*LA.T - A)'

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

'B.is_positive_definite'

True

'LDLB'

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4}-\frac{i}{12} & 1 & 0 \\ \frac{5}{12} & \frac{-\frac{1}{4}+\frac{7i}{12}}{-12\left(\frac{1}{4}-\frac{i}{12}\right)\left(\frac{1}{4}+\frac{i}{12}\right)+2} & 1 \end{bmatrix}$$

$$\begin{bmatrix} 12 & 0 & 0 \\ 0 & -12\left(\frac{1}{4}-\frac{i}{12}\right)\left(\frac{1}{4}+\frac{i}{12}\right)+2 & 0 \\ 0 & 0 & -\frac{\left(-\frac{1}{4}-\frac{7i}{12}\right)\left(-\frac{1}{4}+\frac{7i}{12}\right)}{-12\left(\frac{1}{4}-\frac{i}{12}\right)\left(\frac{1}{4}+\frac{i}{12}\right)+2}+\frac{47}{12} \end{bmatrix}$$

'simplify(LB)'

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4}-\frac{i}{12} & 1 & 0 \\ \frac{5}{12} & -\frac{3}{14}+\frac{i}{2} & 1 \end{bmatrix}$$

'simplify(LB*DB*LB.H - B)'

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

LU разложение

$PA = LU$ для матрицы A

L - левая треугольная матрица с единицами на главной диагонали, U - правая треугольная (трапециевидная) матрица, P - матрица перестановок.

$A = P^{-1}LU.$

Пример 3.

Построим LU разложение для матрицы $$ M=\left(\begin{matrix} -2&3+ I&5 - 2I\

- I&2&1 - I\ 5&-1 + 4I&-3 \end{matrix} \right) $$

In [4]:
```python
M = Matrix([[-2, 3 + I, 5 - 2*I], [-I, 2, 1 - I], [5, -1 + 4*I, -3]])
L, U, perm = M.LUdecomposition()
display(L, U, simplify(expand(L*U)), M, perm)
```

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{i}{2} & 1 & 0 \\ -\frac{5}{2} & \frac{\frac{13}{2}+\frac{13i}{2}}{2-\frac{i(3+i)}{2}} & 1 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 3+i & 5-2i \\ 0 & 2-\frac{i(3+i)}{2} & 1-\frac{i(5-2i)}{2}-i \\ 0 & 0 & \frac{19}{2}-5i-\frac{\left(\frac{13}{2}+\frac{13i}{2}\right)\left(1-\frac{i(5-2i)}{2}-i\right)}{2-\frac{i(3+i)}{2}} \end{bmatrix}$$

$$\begin{bmatrix} -2 & 3+i & 5-2i \\ -i & 2 & 1-i \\ 5 & -1+4i & -3 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 3+i & 5-2i \\ -i & 2 & 1-i \\ 5 & -1+4i & -3 \end{bmatrix}$$

[]

В Примере 3 не пришлось использовать перестановки, параметр perm, описывающий перестановки представляет собой пустой список.

Пример 4.

Заменим в матрице M элемент -2 на 0 и построим LU разложение для новой матрицы.

In [5]:
```python
M[0, 0] = 0
L, U, perm = M.LUdecomposition()
display(L, U, simplify(expand(L*U)), M, perm)
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5i & \frac{(-1-6i)(3-i)}{10} & 1 \end{bmatrix}$$

$$\begin{bmatrix} -i & 2 & 1-i \\ 0 & 3+i & 5-2i \\ 0 & 0 & -3-5i(1-i)-\frac{(-1-6i)(3-i)(5-2i)}{10} \end{bmatrix}$$

$$\begin{bmatrix} -i & 2 & 1-i \\ 0 & 3+i & 5-2i \\ 5 & -1+4i & -3 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 3+i & 5-2i \\ -i & 2 & 1-i \\ 5 & -1+4i & -3 \end{bmatrix}$$

[[0, 1]]

Произведение матриц LU отличается от исходной матрицы M перестановкой строк. Восстановим матрицу M, применяя перестановки в соответствии с результатом, выдаваемым LUdecomposition:

In [6]:
```python
number_of_rows = M.shape[0]
L, U, perm = M.LUdecomposition()
MLU = simplify(expand((L*U).permuteBkwd(perm)))
P = eye(number_of_rows).permuteFwd(perm)
display('L', L, 'U', U, 'perm', perm, 'P', P, 'MLU', MLU,
        'M', M, 'P*M == L*U', P*M == simplify(expand(L*U)))
```

'L'

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5i & \frac{(-1-6i)(3-i)}{10} & 1 \end{bmatrix}$$

'U'

$$\begin{bmatrix} -i & 2 & 1-i \\ 0 & 3+i & 5-2i \\ 0 & 0 & -3-5i(1-i)-\frac{(-1-6i)(3-i)(5-2i)}{10} \end{bmatrix}$$

'perm'

[[0, 1]]

'P'

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

'MLU'

$$\begin{bmatrix} 0 & 3+i & 5-2i \\ -i & 2 & 1-i \\ 5 & -1+4i & -3 \end{bmatrix}$$

'M'

$$\begin{bmatrix} 0 & 3+i & 5-2i \\ -i & 2 & 1-i \\ 5 & -1+4i & -3 \end{bmatrix}$$

'P*M == L*U'

True

LU разложение можно применять и для прямоугольной матрицы.

Пример 5.

Добавим к матрице M справа столбец из чисел 1, 2, 3 и построим LU разложение для новой матрицы.

In [7]:
```python
number_of_rows = M.shape[0]
M = M.row_join(Matrix([k + 1 for k in range(number_of_rows)]))
L, U, perm = M.LUdecomposition()
MLU = simplify(expand((L*U).permuteBkwd(perm)))
P = eye(number_of_rows).permuteFwd(perm)
display('L', L, 'U', U, 'perm', perm, 'P', P, 'MLU', MLU,
        'M', M, 'P*M == L*U', P*M == simplify(expand(L*U)))
```

'L'

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5i & \frac{(-1-6i)(3-i)}{10} & 1 \end{bmatrix}$$

'U'

$$\begin{bmatrix} -i & 2 & 1-i & 2 \\ 0 & 3+i & 5-2i & 1 \\ 0 & 0 & -3-5i(1-i)-\frac{(-1-6i)(3-i)(5-2i)}{10} & 3-10i-\frac{(-1-6i)(3-i)}{10} \end{bmatrix}$$

'perm'

[[0, 1]]

'P'

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

'MLU'

$$\begin{bmatrix} 0 & 3+i & 5-2i & 1 \\ -i & 2 & 1-i & 2 \\ 5 & -1+4i & -3 & 3 \end{bmatrix}$$

'M'

$$\begin{bmatrix} 0 & 3+i & 5-2i & 1 \\ -i & 2 & 1-i & 2 \\ 5 & -1+4i & -3 & 3 \end{bmatrix}$$

'P*M == L*U'

True

QR разложение

$A = QR$ для симметричной вещественной матрицы A

Q - матрица из ортогональных столбцов, т.е. $Q^HQ = I$, I - единичная матрица, причем может не выполняться $QQ^H = I$ (для ортогональной матрицы $Q^HQ = QQ^H = I$), R - правая треугольная (трапециевидная) матрица.

Ранг матрицы A равен числу столбцов матрицы Q.

Пример 6.

Построим QR разложение для матрицы Примера 5.

In [8]:
```python
Q, R = M.QRdecomposition()
MQR = simplify(expand(Q*R))
display('Q', simplify(expand(Q)), 'R', simplify(expand(R)),
        'MQR', MQR, 'M', M, 'M = QR', M == MQR)
```

'Q'

$$\begin{bmatrix} 0 & \frac{\sqrt{858}(3+i)}{99} & \frac{\sqrt{957}(28+17i)}{2871} \\ -\frac{\sqrt{26}i}{26} & \frac{5\sqrt{858}(6-i)}{2574} & -\frac{5\sqrt{957}(17+i)}{2871} \\ \frac{5\sqrt{26}}{26} & -\frac{\sqrt{858}(1+6i)}{2574} & \frac{\sqrt{957}(-1+17i)}{2871} \end{bmatrix}$$

'R'

$$\begin{bmatrix} \sqrt{26} & \frac{\sqrt{26}(-5+22i)}{26} & \frac{\sqrt{26}(-14+i)}{26} & \frac{\sqrt{26}(15+2i)}{26} \\ 0 & \frac{3\sqrt{858}}{26} & \frac{\sqrt{858}(376-329i)}{2574} & \frac{\sqrt{858}(135+2i)}{2574} \\ 0 & 0 & \frac{\sqrt{957}}{99} & -\frac{\sqrt{957}(5+2i)}{99} \end{bmatrix}$$

'MQR'

$$\begin{bmatrix} 0 & 3+i & 5-2i & 1 \\ -i & 2 & 1-i & 2 \\ 5 & -1+4i & -3 & 3 \end{bmatrix}$$

'M'

$$\begin{bmatrix} 0 & 3+i & 5-2i & 1 \\ -i & 2 & 1-i & 2 \\ 5 & -1+4i & -3 & 3 \end{bmatrix}$$

'M = QR'

True

Решение систем линейных уравнений с помощью разложений.

Пример 7.

Решим с помощью QR разложения матрицы

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

систему линейных уравнений

$$BX = b, \quad b = \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix}.$$

Проверим совместность СЛАУ:

In [9]:
```python
B = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b = Matrix([6, 12, 24])
Bb = B.row_join(b)
print('B.rank =', B.rank(), 'Bb.rank =', Bb.rank(),
      'B.rank == Bb.rank', B.rank() == Bb.rank())
```

```
B.rank = 2 Bb.rank = 3 B.rank == Bb.rank False
```

СЛАУ несовместна, в обычном смысле решения нет, но с помощью QR разложения можно найти псевдорешение, т.е. такую матрицу- столбец, что при подстановке в СЛАУ вместо X даст минимальную возможную норму разности левой и правой частей СЛАУ (невязки).

In [10]:
```python
X = B.QRsolve(b)
display(X)
```

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Для сокращения объема используемой памяти используется сокращенная форма QR разложения, при которой прямоугольная часть матрицы Q, состоящая из одних нулей, не хранится. Для восстановления полного решения в нашем случае достаточно добавить один ноль к получившемуся столбцу (поскольку столбце X должен состоять из трех элементов):

In [11]:
```python
X = X.col_join(Matrix([0]))
delta = B*X - b
display(X, delta, delta.norm(2))
```

$$\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

$\sqrt{6}$

Полученное псевдорешение не является нормальным псевдорешением, т.е. псевдорешением с минимальной нормой, но и нормальное псевдорешение проще получить, используя QR разложение.

Жорданова форма матрицы

$A = PJP^{-1}$ для квадратной матрицы A

P - матрица перехода, J - жорданова матрица.

Пример 8.

Построим жорданову форму для матриц

$$B = \begin{pmatrix} 6 & 5 & -2 \\ -3 & -1 & 3 \\ 2 & 1 & -2 \end{pmatrix} \quad K = \begin{pmatrix} 6 & 5 & -2 & -3 \\ -3 & -1 & 3 & 3 \\ 2 & 1 & -2 & -3 \\ -1 & 1 & 5 & 5 \end{pmatrix}$$

In [12]:
```python
B = Matrix([[6,  5, -2], [-3, -1,  3], [2,  1, -2]])
P, J = B.jordan_form()
P = simplify(expand_complex(expand(P)))
display('P', simplify(expand(P)), 'J', simplify(expand(J)),
        'PJP**(-1)', P*J*P**(-1), 'B', B)
```

'P'

$$\begin{bmatrix} 1 & 9 & 1 \\ -1 & -6 & 1 \\ 1 & 3 & 0 \end{bmatrix}$$

'J'

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

'PJP**(-1)'

$$\begin{bmatrix} 6 & 5 & -2 \\ -3 & -1 & 3 \\ 2 & 1 & -2 \end{bmatrix}$$

'B'

$$\begin{bmatrix} 6 & 5 & -2 \\ -3 & -1 & 3 \\ 2 & 1 & -2 \end{bmatrix}$$

In [13]:
```python
K = Matrix([[6,  5, -2, -3], [-3, -1,  3,  3], [2,  1, -2, -3], [-1,  1,  5,  5]])
P, J = K.jordan_form()
display('P', simplify(expand(P)), 'J', simplify(expand(J)),
        'PJP**(-1)', P*J*P**(-1), 'K', K)
```

'P'

$$\begin{bmatrix} 4 & 1 & 5 & 0 \\ -3 & 0 & -3 & 1 \\ 2 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 \end{bmatrix}$$

'J'

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

'PJP**(-1)'

$$\begin{bmatrix} 6 & 5 & -2 & -3 \\ -3 & -1 & 3 & 3 \\ 2 & 1 & -2 & -3 \\ -1 & 1 & 5 & 5 \end{bmatrix}$$

'K'

$$\begin{bmatrix} 6 & 5 & -2 & -3 \\ -3 & -1 & 3 & 3 \\ 2 & 1 & -2 & -3 \\ -1 & 1 & 5 & 5 \end{bmatrix}$$