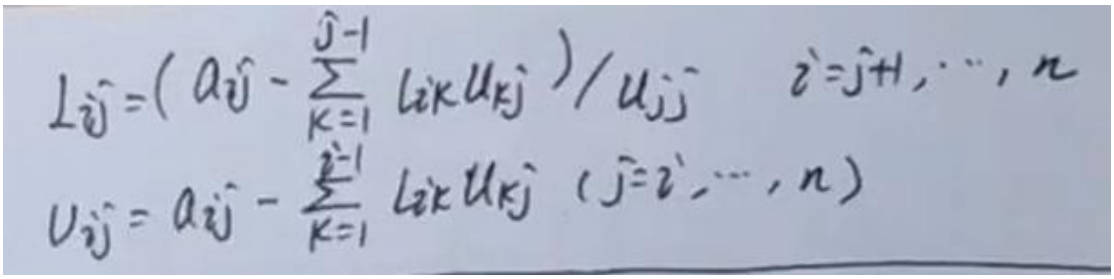
根据 LU 分解公式：



可得代码：

```
def getLU(A):
    shape=A.shape
    L=np.zeros(shape)
    U=np.zeros(shape)
    r,_=shape

    for i in range(r):
        for j in range(r):
            sum=0
            for k in range(i):
                sum+=L[i,k]*U[k,j]
            U[i,j]=A[i,j]-sum
        for j in range(r):
            sum=0
            for k in range(i): # 求的是j i
                sum+=L[j,k]*U[k,i]
            L[j,i]=(A[j,i]-sum)/U[i,i]


    return L,U
```

根据笔算过程可进行优化，

1. U 的第一行可直接写出
2. L 的第一列可以由 U 的第一行简单计算得出
3. L 的上三角部分直接写 0，对角线直接写 1
4. U 的下三角部分直接写 0

优化后代码：

```
def getLU(A):
    shape=A.shape
    L=np.zeros(shape)
    U=np.zeros(shape)
    r,_=shape

    for i in range(r):
        U[0,i]=A[0,i]
    for i in range(r):
        L[i,0]=A[i,0]/U[0,0]
```

```
    for i in range(1,r):
        for j in range(r):
            if j < i:
                U[i, j] = 0
            else:
                sum=0
                for k in range(i):
                    sum+=L[i,k]*U[k,j]
                U[i,j]=A[i,j]-sum
        for j in range(r):
            if i==j:
                L[j,i]=1
            elif j < i: # 注意这里 i 和 j 是反的
                L[j,i]=0
            else:
                sum=0
                for k in range(i): # 求的是 j i
                    sum+=L[j,k]*U[k,i]
                L[j,i]=(A[j,i]-sum)/U[i,i]

    return L,U
```

代入数据：

```
A=[[4,2,0,0],[1,4,1,0],[0,1,4,1],[0,0,2,4]]
A=np.array(A)
L,U=getLU(A)
```

可得结果 L、U 分别为：

```
[[1.          0.          0.          0.        ]
 [0.25        1.          0.          0.        ]
 [0.          0.28571429  1.          0.        ]
 [0.          0.          0.53846154  1.        ]]

[[4.          2.          0.          0.        ]
 [0.          3.5         1.          0.        ]
 [0.          0.          3.71428571  1.        ]
 [0.          0.          0.          3.46153846]]
```

由对角为 1 的下三角阵（L）快速解方程代码如下：

```
def getY(L,b): # 使用行向量
    r=b.shape[0]
    y=np.zeros(b.shape)
    y[0]=b[0]
    for i in range(1,r):
        y[i]=b[i]
```

```
        for j in range(i):
            y[i]-=L[i,j]*y[j]
    return y
```

代入数据：

```
b=[-1,0,0,0]
b=np.array(b)
y=getY(L,b)
```

可得 y：

[-1. 0.25 -0.07142857 0.03846154]

由上三角阵（U）快速解方程的代码如下：

```
def getX(U,b):
    r = b.shape[0]
    y = np.zeros(b.shape)
    y[r-1] = b[r-1] / U[r-1,r-1]
    for i in range(r-2, -1, -1): # 最后一行完事了，从倒数第二行开始
        y[i] = b[i]
        for j in range(r-1, i, -1): # 从最后一个往前减，i 不减，直接除
            y[i] -= U[i, j] * y[j]
        y[i] /= U[i, i]
    return y
```

代入数据：

```
x=getX(U,y)
```

可得最终结果 x：

[-0.28888889 0.07777778 -0.02222222 0.01111111]