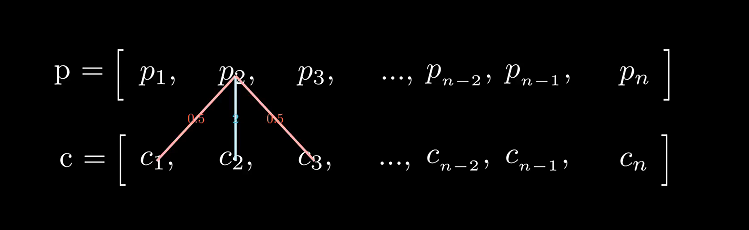
题目翻译：

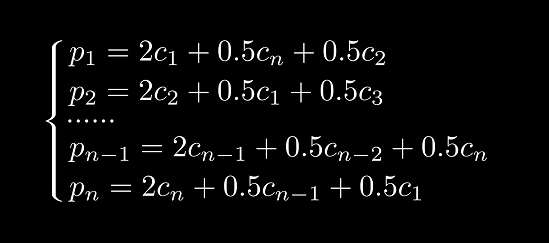


给定两个数列 数列p和数列c

这两个数列满足这样的关系： 数列p中的每一个元素 = 数列c中对应位置的元素的二倍+c中两边的元素之和的一半

每个位置都满足， 以此类推

那么我们就可以根据这样的关系列出方程组

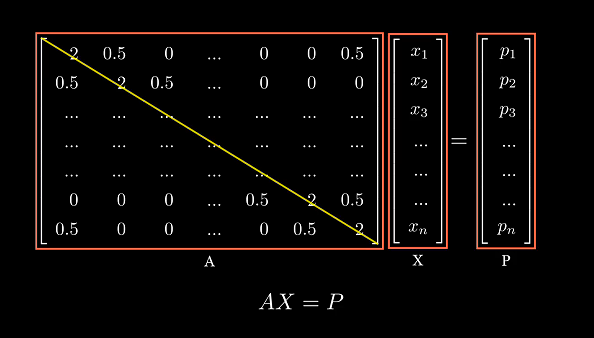


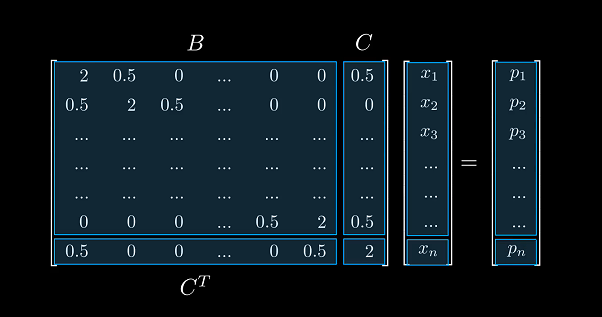
再把方程组写成矩阵乘法的形式 长这样

这是矩阵A 这是列向量X， 这是列向量P

而且我们发现 矩阵A是一个三对角矩阵， 即：只有主对角线 以及 主对角线左边右边的元素不为0， 其他都为0

用矩阵乘法的形式表示 就是ax=p



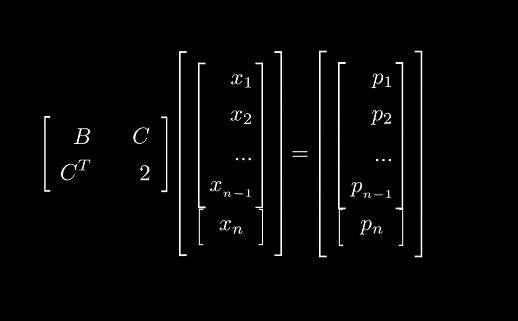


接着 我们对矩阵A进行分块

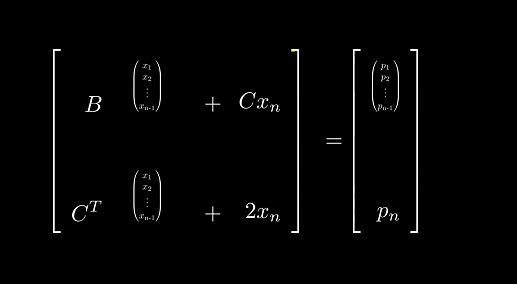
如果说矩阵A是n阶方阵 那么将左上角的这n-1阶设为矩阵B， 右边的这个长度为n-1的列向量设为向量C， 下面的这个行向量设为c转置

接着 同样的 将向量X与向量P划分成长度为n-1和1的两部分

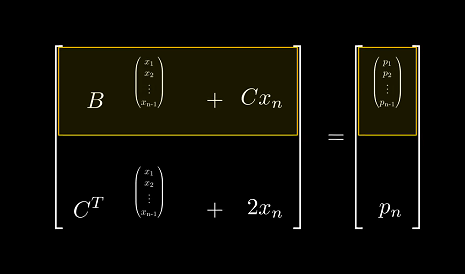
也就是这样：



接着 看等号左边 把分块矩阵看做一个整体 看做是矩阵元素也按照矩阵乘法的法则进行相乘 那就变成这样



即- 把等号左边看做是2x2方阵 乘以 2x1矩阵 得到2行1列的矩阵



现在在等号两边 都是2行1列的分块矩阵

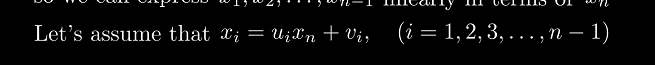
接着对上半部分列一个等式

我们观察这个方程的形式 一个常数矩阵B 乘以一个我们要求解的列向量 x\_1 到 x\_n-1, 加上一个常数C乘以未知数x\_n, 最后得到系数矩阵p1 到 p\_n

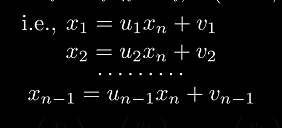
注意到有n个未知数

所以 为了求解这个等式 我们不妨先将x\_1 - x\_n-1 用x\_n线性表示，

即： 我们设:

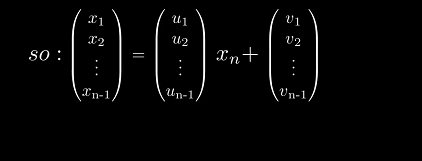


也就是说：



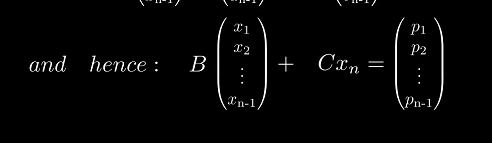
观察到每个式子都有x\_n 所以用向量和常数数乘的方式，得到这个用向量表示的式子

x向量 等于 u向量数乘xn 加 v向量

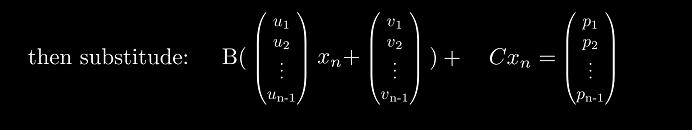


然后别忘了 我们刚刚用矩阵乘法推出的这个等式：

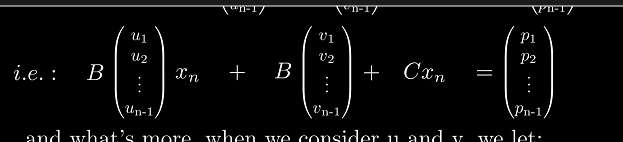
即 常数矩阵b 乘以x向量 加常数C 乘以xn 等于 b向量



我们把x向量 用u乘以x加v带入

得到：

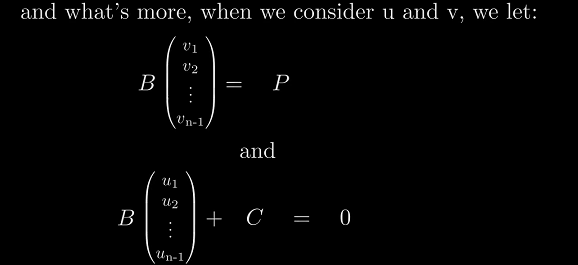
由于数乘有分配率 自然把他展开

即 我们可以得到：

此外 我们刚刚在设u和v的时候 也不是随便设的

在设系数向量u和v的时候 我们另u和v满足： 矩阵B乘以向量v等于向量p

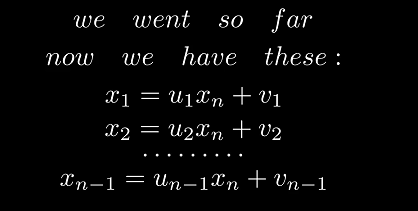
矩阵B乘以向量u 加上常数C等于0



（因为我们刚刚在设uv的时候 很显然 每一个uv都有无限组解 所以取这个特殊的情况当然是可以的）

所以我们就把原问题转换成了 求解这两个 三对角线性系统

接着再回过头来看我们有的东西：



X1 - xn-1 用xn线性表示，

然后再根据原方程的最后一行 xn也是可以用x1 和xn-1求解出来的

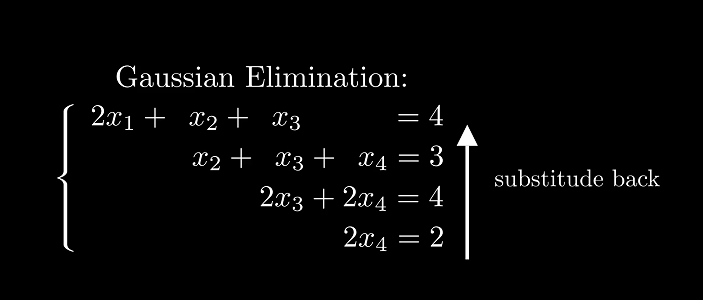
所以现在的问题是 如何求解向量u和向量v呢？

对于计算机而言 有一个可以用O(n)时间复杂度和O(n)空间复杂度的解法 来求解这个三对角矩阵： Crout分解

我们先从最基础的解方程开始：

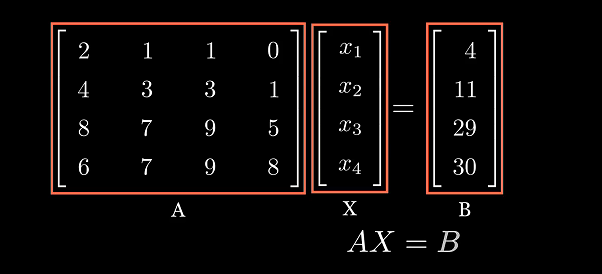
早在小学的时候 老师就教过我们 解方程的一个重要的思路是消元

我们可以通过这样 逐行消掉一些未知数的方式 简化方程 然后从下往上回代 求出方程的解



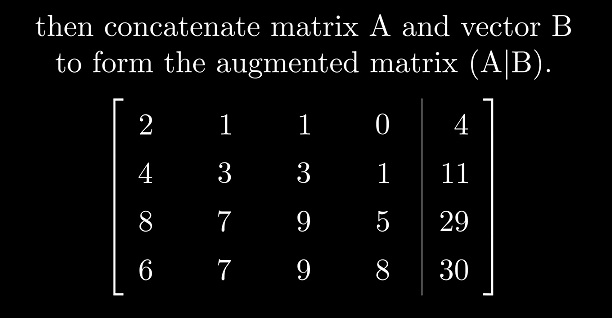
在大学的线性代数课堂中 老师也会讲到类似的方法 只不过这时候是用矩阵的形式来表示方程组：

首先还是一样 用矩阵乘法的方式 表示出这个方程组：



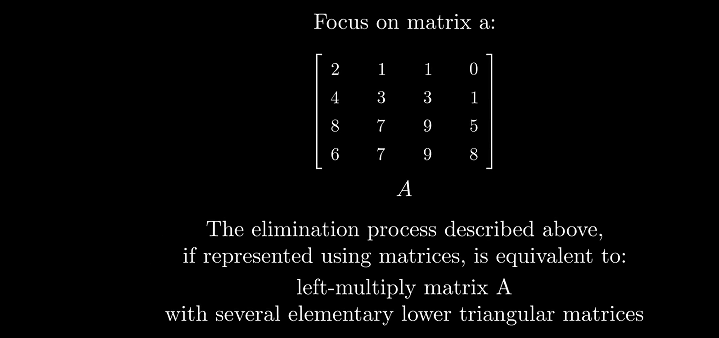
系数矩阵A， 解向量X， 常数向量B

然后把A和B拼在一起形成增广矩阵AB



接着我们来关注矩阵A

刚刚的逐行消元的动作 我们进行的操作是： 将矩阵A左乘上一些初等下三角矩阵



左乘矩阵L1 然后消掉第一列的下面三个元素

左乘矩阵L2 然后消掉第二列的下面二个元素

左乘矩阵L3 然后消掉第三列的下面一个元素

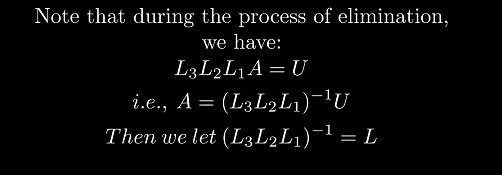
与此同时呢 在初等行变换的过程中呢 对右边的常数向量也做同样的操作（也就是随着左边的系数矩阵做同样的行变换） 这样我们就得到一个新的增广矩阵UC

这两种过程 本质上是相同的

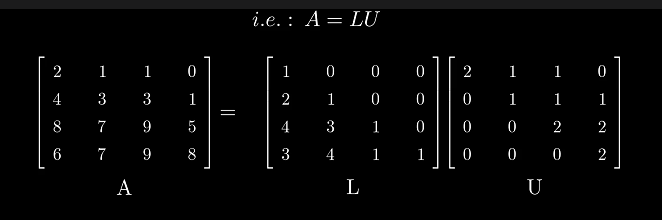
在上述的消元过程中 我们有 这样一个等式： 矩阵A 依次左乘L1 L2 L3 得到矩阵U

那么 我们可以把L1L2L3放到等号右边去， 也就是 A = L1L2L3逆 U,

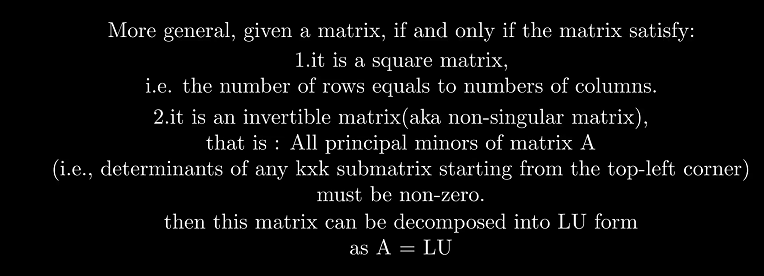
然后 我们不妨设 L1L2L3的逆等于L



即： A = LU



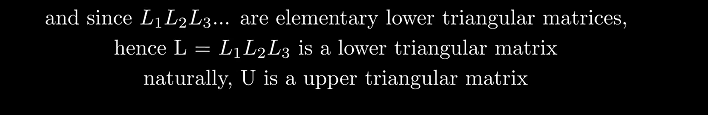
接下来再推广到一般情况：



如果一个矩阵满足：

1. 这是一个方阵
2. 这是一个可逆矩阵

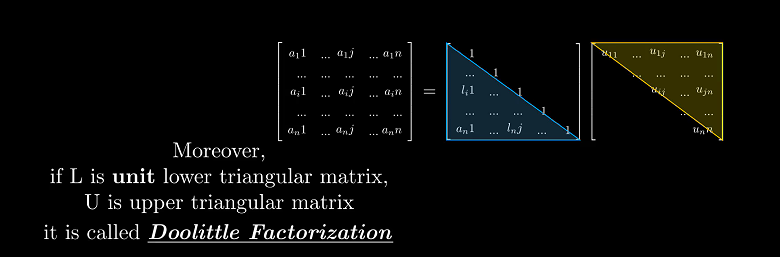
那么这个矩阵就可以被分解成 A = LU的形式



然后因为l1l2l3都是初等下三角矩阵

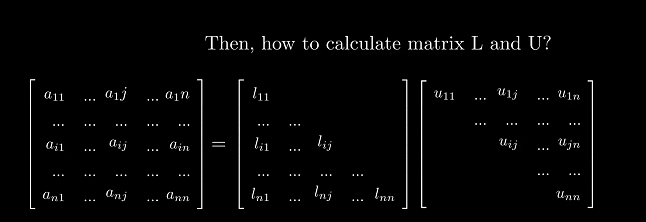
所以L = l1l2l3逆 也是个下三角矩阵

自然而然的 U是一个上三角矩阵



更多的， 如果L是**单位**下三角矩阵， U是上三角矩阵， 那么这个过程就叫做Doolittle分解，

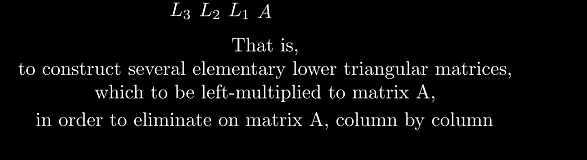
反之， 如果L是下三角矩阵， U是**单位**上三角矩阵， 那么这个过程就叫做Crout分解

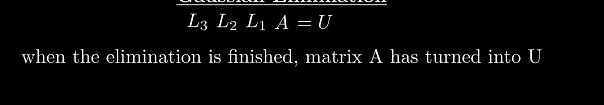


接下来的问题是： 如何计算矩阵L和矩阵U呢？

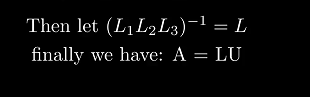
在上文中 我们已经提到了一个方法： 高斯消元。

即 构造几个初等下三角矩阵 去左乘到矩阵A， 为了去在矩阵A上 做逐列的消元





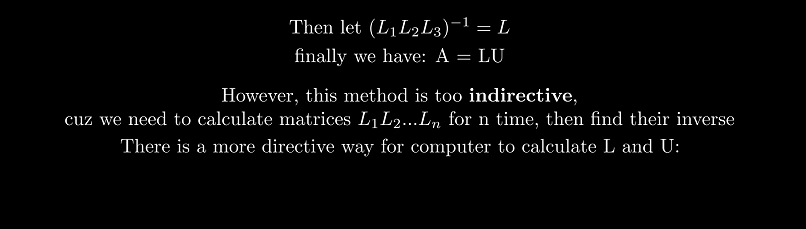
当消元结束了 矩阵A就变成了U



然后再让l1l2l3逆等于L

最后我们就得到了 A = LU

然而呢 这种方法非常的麻烦

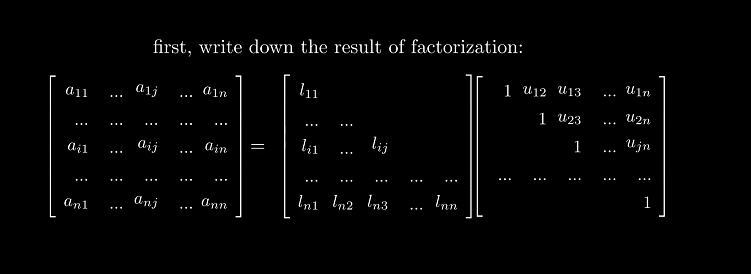


因为我们需要去计算n次 初等下三角矩阵 l1,l2, ..., ln-1 一共n次 去找到他们的逆矩阵

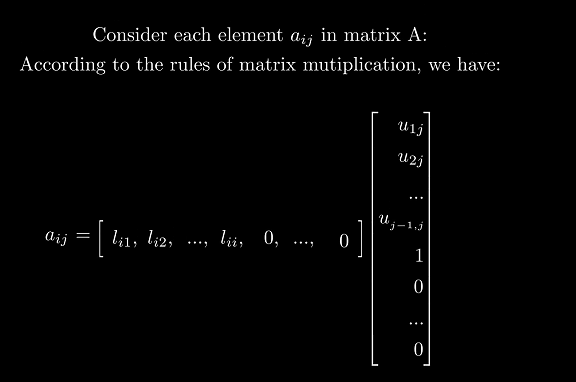
这种方式 适合小规模的矩阵 在草稿纸上手酸，

但是 还有一种更适合计算机的方式 能够更直接的计算出矩阵L和U：

首先 写下分解的结果：



然后从结果出发 根据矩阵的乘法法则 倒推出LU

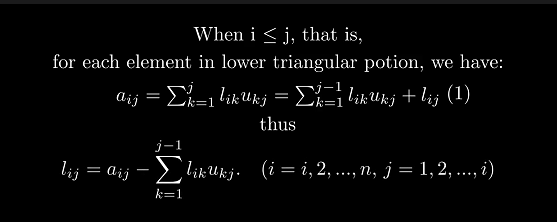


对于矩阵A中的任意一个元素aij 根据矩阵乘法的法则 也就是 rows by colunm的法则， 我们有：

Aij 等于这样的一个行向量 乘以这样一个列向量

在行向量中 有i个非零元素

在列向量中 有j个非零元素



当i>=j的时候， 即： 对于下三角矩阵的每一个元素 都有这样的一个等式 也就是对应的l与u相乘， 然后由于i比j大， 我们把那个1乘以lij单独拿出来 再做个变形 就得到了lij的这个式子 也就是矩阵j的计算方式

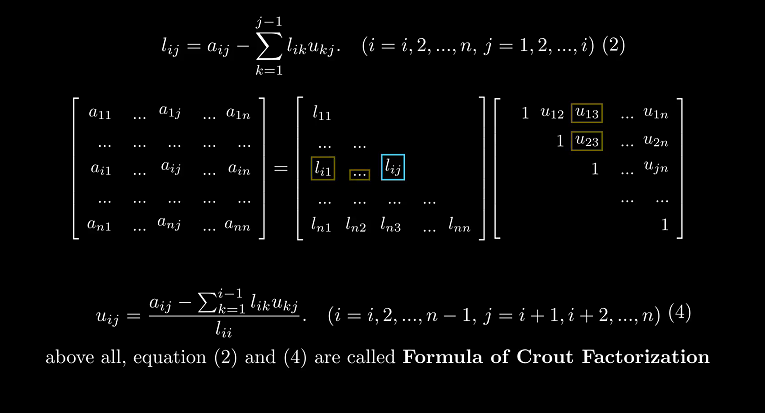
同理 可以得到矩阵u的计算方式， 把这个sigma符号中的最后一项单独拿出来 放到等号的一边

这个公式2与公式4 被称为Crout分解公式

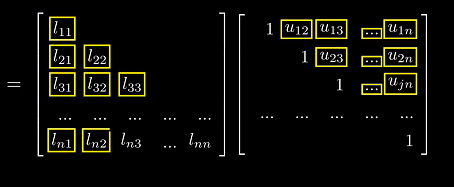
同理可得Doolittle分解公式如下：

更具体一点 如何通过这种方式 计算crout分解的结果呢？

我们不难发现 要计算矩阵l中的的某个元素 需要先计算l中的同一行的左边的元素 以及u中的同一列的 这个位置的上方的元素



所以我们可以采取像这样的逐行递推的方式：



先计算第一行 l矩阵第一行的一个 u矩阵第一行的n-1个

然后第二行 l矩阵第二行的二个 u矩阵第二行的n-2个

。。。以此类推 最后一行

l矩阵的n-1个

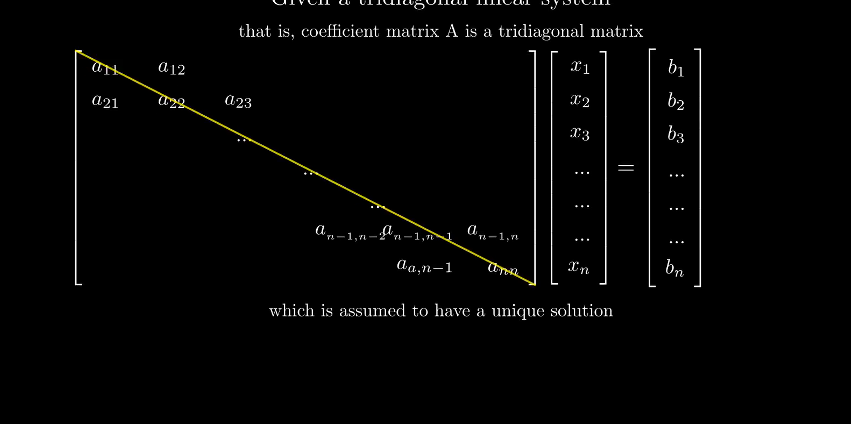


更具体的 如果系数矩阵A是一个三对角矩阵

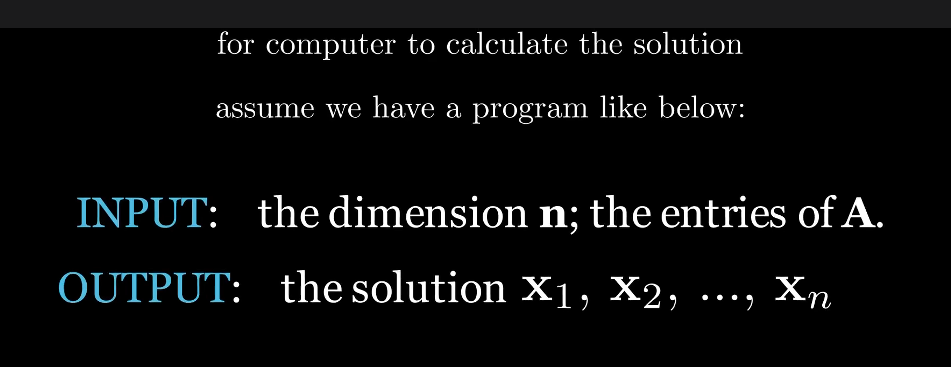
并且我们假设他有唯一解

我们可以设计这样一个程序 使得能让计算机更方便的计算出结果

时间复杂度和空间复杂度都是on



假设我们有这样一个程序

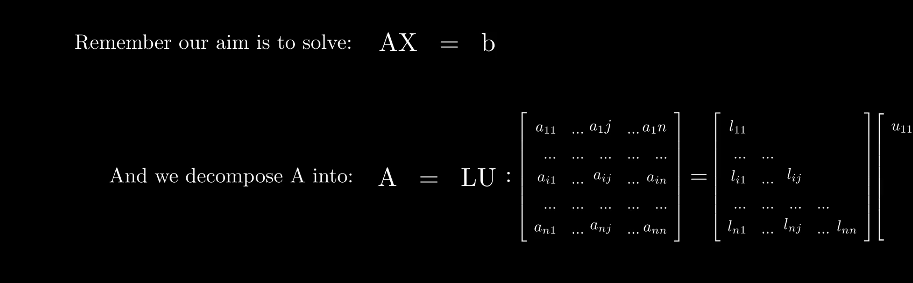


输入是矩阵的维度 系数矩阵A

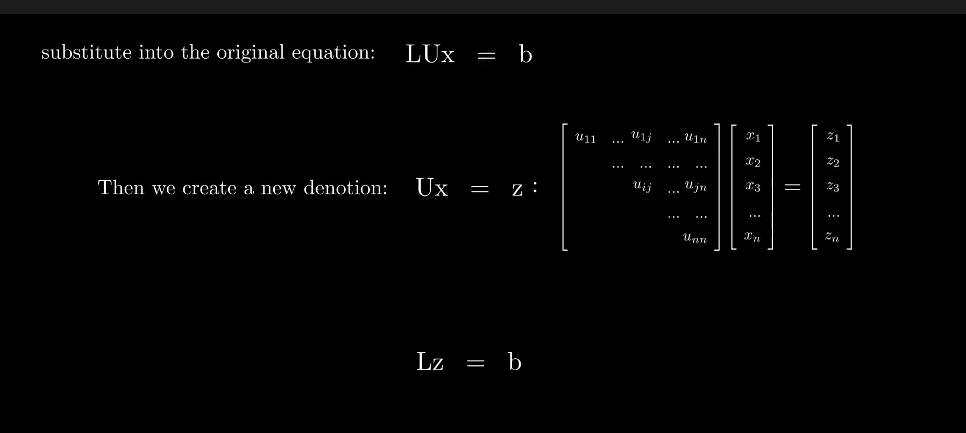
输出是方程的解

到这里 我们先不急往下看。 我们先理一下思路

别忘了我们的原始目标是求解ax=b



然后我们把a分解成lu



然后带入到原方程中 得到lux = b

接着我们令 ux = z

所以 原方程就被转化成了 求解lz = b

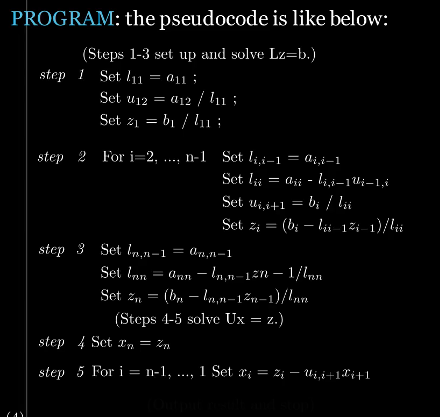
求出z以后 再求解x即可

再结合我们之前推导出的crout分解的公式

、

我们可以设计这样一个程序：

伪代码如下：



我就不说具体过程了， 看我动画演示计算顺序。 看方框框出的元素就是计算的顺序。

综上所述 所以程序可以这么设计：

*第一步到第三步 计算 求解方程lz = b*

*首先第一步 初始化 求出l矩阵第一行的 对角线上的元素， 以及u矩阵对角线右侧的元素， 然后求出z向量的第一个元素*

*然后i从2循环到n-1 ， 按照顺序计算出 l矩阵的所有对角线上的元素， 然后计算出u矩阵所有的 对角线右侧的元素，*

最后计算出l矩阵最后一行 对角线上的元素 u矩阵对角线右侧的元素， 以及z向量的最后一个元素

求解出了z向量， 接下来求解：ux = z

首先 我们有xn = zn， 即 x向量的最后一个元素就等于z向量的最后一个元素，接着 i从n-1循环到1， 分别求解出x。 然后提交代码，过啦！ 大功告成。

当然了 这题的解法不唯一。 这个解法只是我搜博客搜到的一个“看起来挺官方的”解法。

本质上这还是一个解方程题目。 如果你能想到一个方法 能在给定的时间和空间内 做到消元 求解方程 一样都是能通过的~