Matrizes tridiagonais, tridiagonais cíclicas e sistemas.

Decomposição LU para Matrizes Tridiagonais

Marco Antônio Rudas Napoli, n°USP: 11857970

Circuito eletrônico em fundo preto

Descrição gerada automaticamente com confiança média

**Decomposição LU**

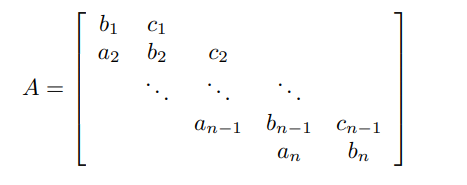
Certas matrizes podem ser triangularizadas pelo Método de Eliminação de Gauss sem que haja troca de linhas. Assim, conseguimos montar duas matrizes, L e U, a partir de uma matriz A.

A matriz L é triangular inferior com os itens da diagonal principal igual a 1 e com os demais iguais aos multiplicadores resultantes do Método de Eliminação de Gauss.

Enquanto isso, a Matriz U é simplesmente a matriz A escalonada.

Assim, conseguimos montar a relação A = LU.

**Decomposição LU de uma matriz tridiagonal:**

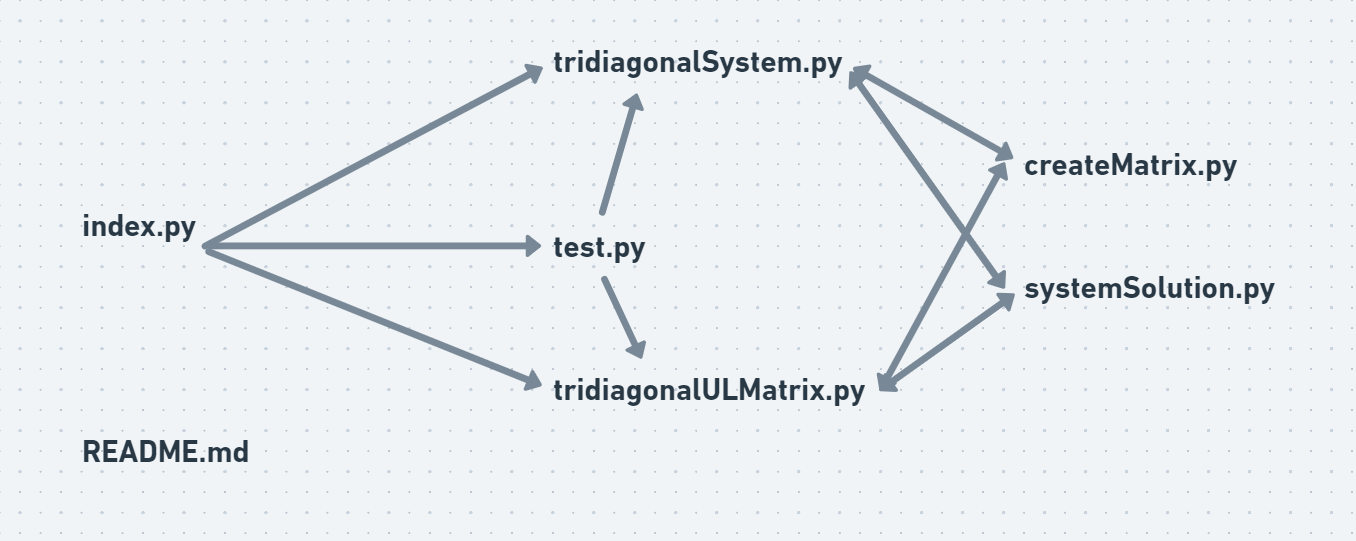
Sabemos que uma matriz tridiagonal possui o seguinte formato:

Ao fazermos a decomposição, encontramos que a matriz L possui itens apenas na

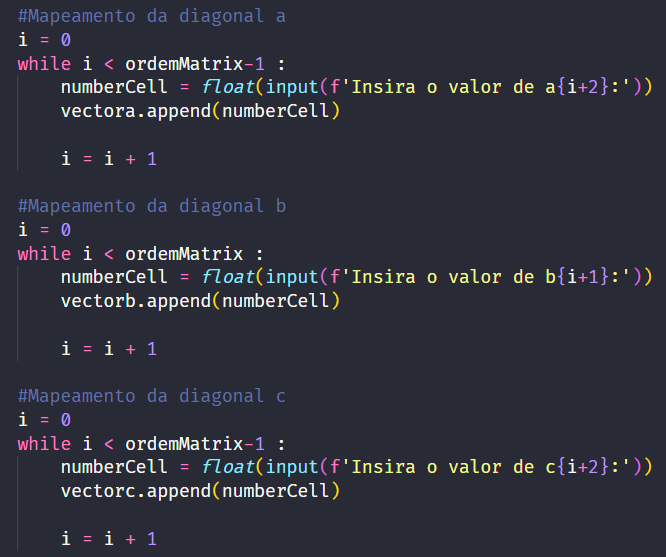
diagonal secundária inferior. A matriz U possui a diagonal principal seguindo o Método de Eliminação de Gauss e a diagonal secundária superior como os mesmos itens da Matriz A, enquanto todos os outros tem valor igual a “0”.

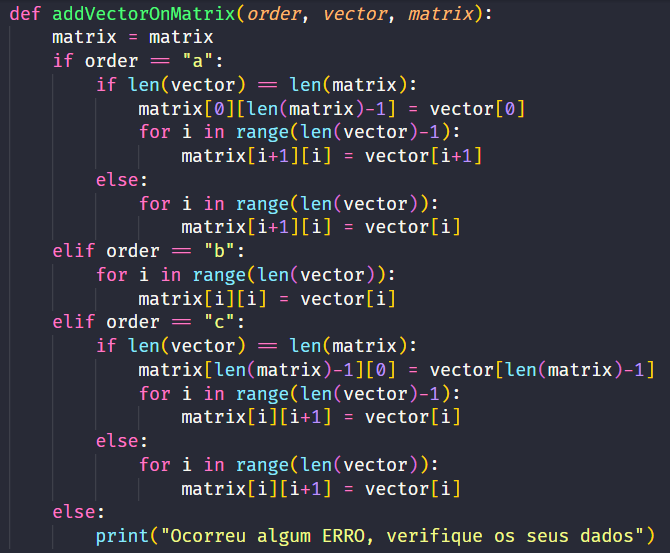
Dessa maneira podemos economizar processamento e armazenar os nossos dados de maneira mais inteligente.

**Estrutura de Funções do Projeto:**

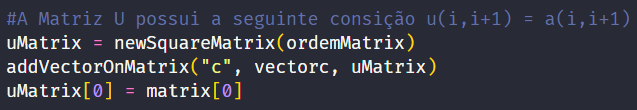
****

**Solução 1 - Decomposição LU de Matriz Tridiagonal**

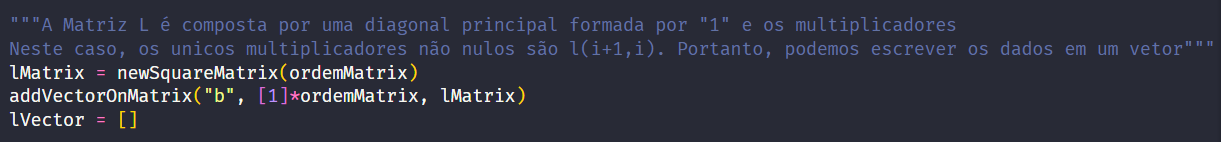
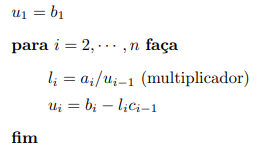
Após introduzirmos os valores dos vetores a, b e c a partir do seguinte código podemos iniciar o algorítimo para achar o as matrizes L e U.

A partir disso, introduzimos os valores mapeados numa matriz, criada pela da função “newSquareMatrix(ordemMatrix)”, utilizando a função “addVectorOnMatrix(order, vector, matrix)”. (obs: O algorítmo desta última foi desenvolvido de maneira a atender o segundo exercício proposto).

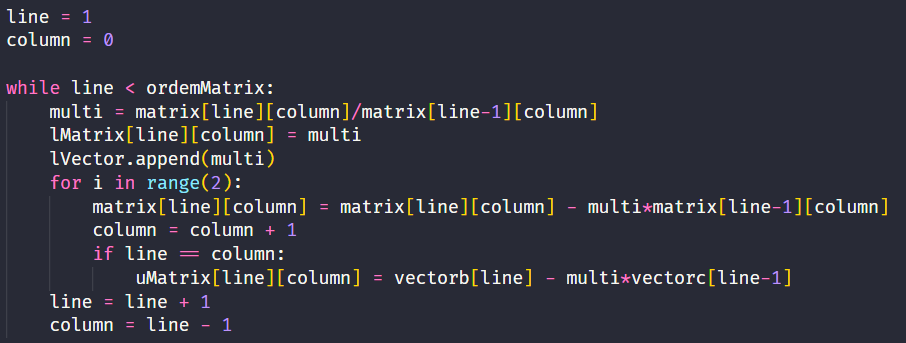
Possuindo a matriz A em mão já podemos começar a trabalhar para acharmos as matrizes U e L.

Sabemos que a matriz U possui a seguinte propriedade: ui, i+1 = ai, i+1. Assim temos que, a matriz U, inicialmente formada por zeros, terá sua matriz secundária superior identica à matriz A, economizando processamento para sua construção. Além disso, sabemos que a primeira linha é identica a da matriz A.

Seguindo para a construção da matriz L, temos que os únicos itens que não são obrigatoriamente zeros são pertencentes à matriz principal e à secundária inferior. Tendo essa propriedade em mão, conseguimos dar o formato da matriz da seguinte forma:

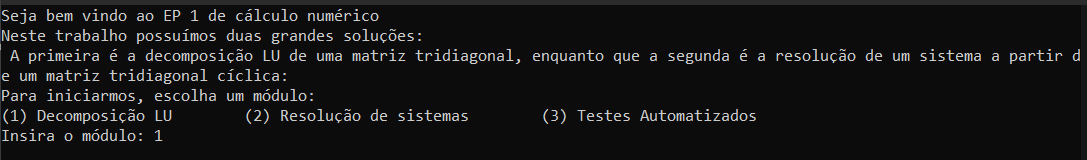


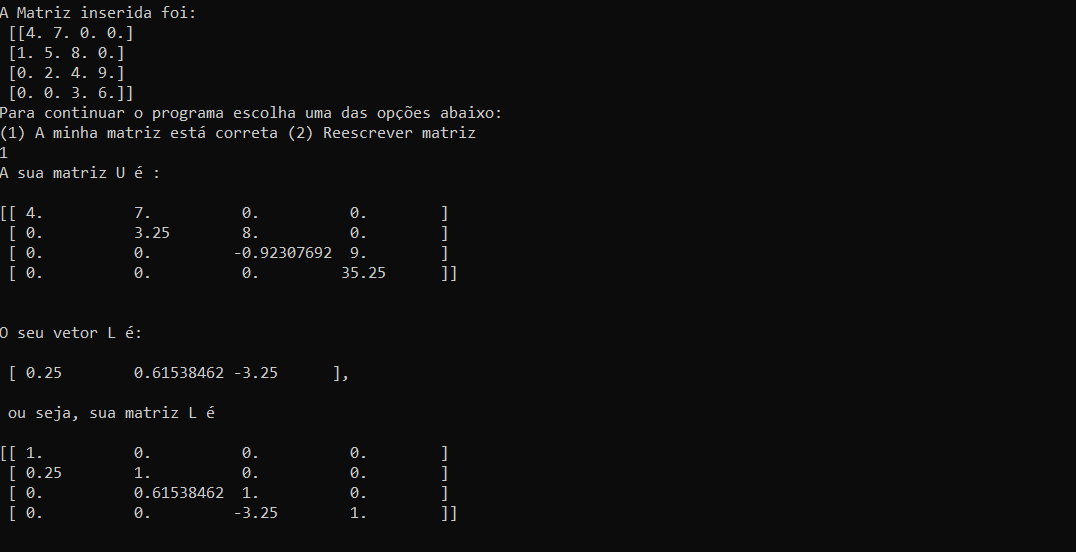
Agora que já possuímos tanto o formato da matriz U e L, quanto a matriz A, podemos começar a rodar o algorítmo apresentado no enunciado deste EP, que é:

Por ser uma matriz tridiagonal, não há a necessidade de passar por todos os itens da matriz para calcularmos os multiplicadores, apenas aqueles que não possuem 0 no item acima. Além disso, conseguimos utilizar os vetores a, b e c para melhorar a intermpretação dos cálculos. Logo, o código para acharmos a matriz L e U possue a seguinte estrutura:

Observe que sempre estamos caminhando uma coluna a menos quando comparado à linha, isso porque não faz sentido passar pelos itens que já foram zerados.

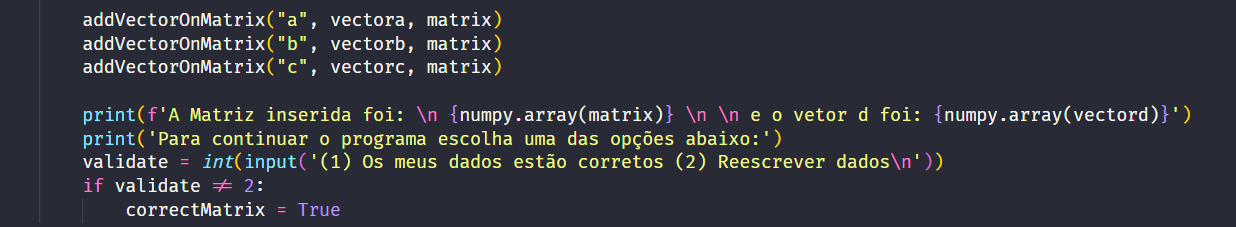
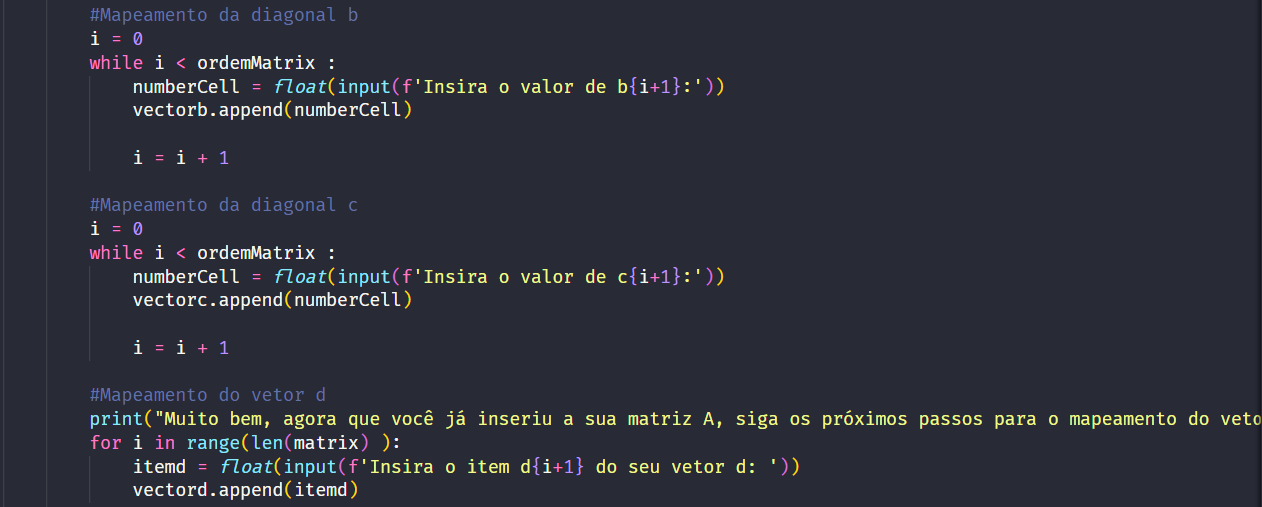
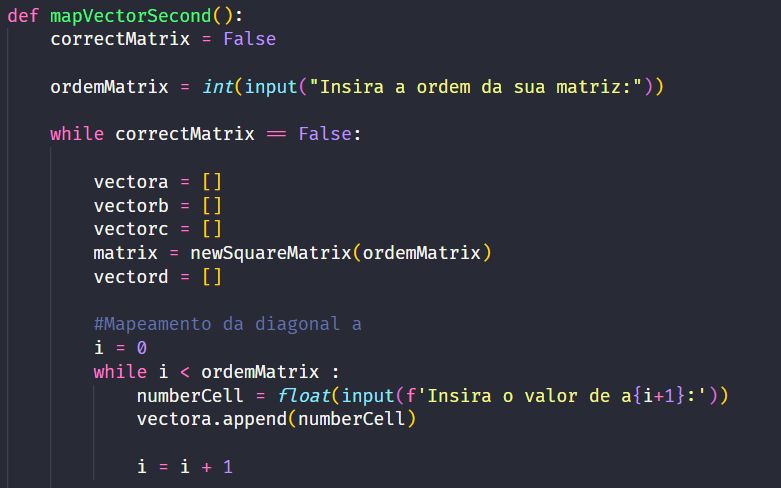
Por fim, aplicando esse algorítmo chegamos no resultado final, com a função retornando tanto a matrix U quando a L.

Segue abaixo um exemplo de funcionamento da primeira solução: Texto

Descrição gerada automaticamente

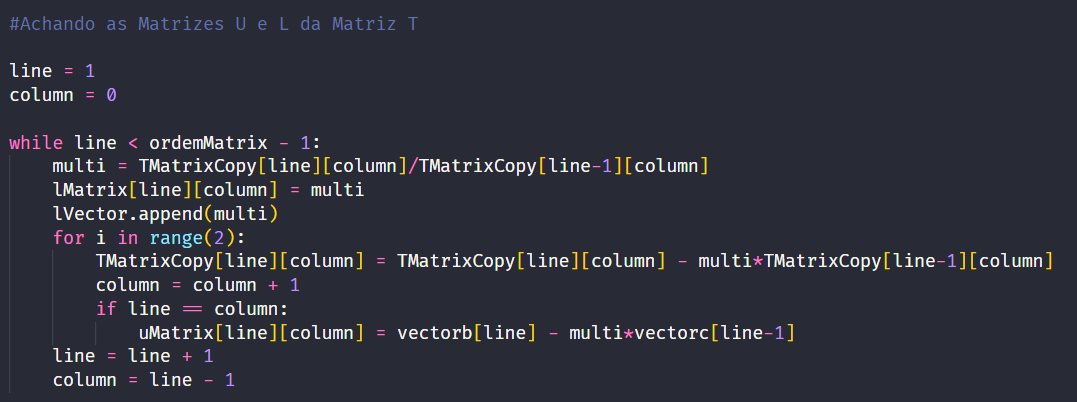
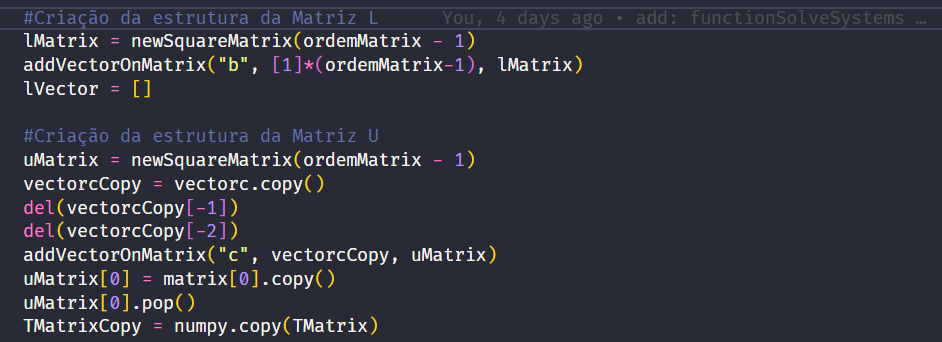
**Solução 2 - Resolução de sistemas:**

A solução dois, por sua vez, é dependente do entendimento da solução 1, já que é necessário encontrarmos a matriz U e L para resolvermos um sistema tridiagonal cícliclo.

Temos o mapeamento dos itens da matriz A e do vetor d a partir da função “mapVectorSecond()” mostrada a seguir:

Após mapearmos os vetores e adicionarmos na Matriz A é necessário acahrmos a subMatriz T de tamanho (n-1)x(n-1) para que possamos solucionar o sistema. Para isso, basta excluirmos as ultimas linha e coluna da matriz A.

(obs: armazenamos a ultima coluna sem o ultimo item um vetor chamado de v e, análogamente, armazenamos a ultima linha sem o último item num vettor denominado de w)

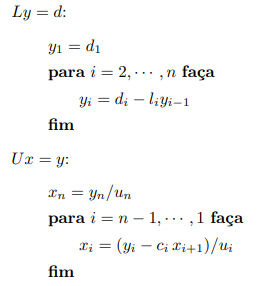
A partir do momento que possuimos a matriz T, é ncessário encontrarmos suas respectivas matrizes U e L, ou seja, conseguimos reutilizar o algorítmo feito no primeiro exercício, obviamente tratando os novos dado

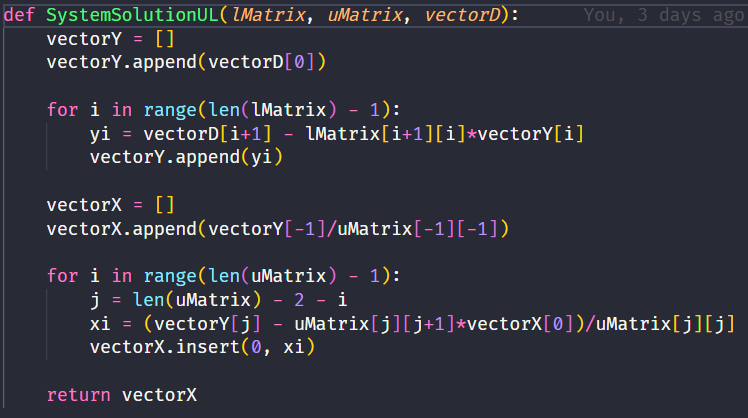
Obtendo as respectivas matrizes L e U partimos para o algorítmo passado no EP para resolução de sistemas. Esta tarefa tem como objetivo encontrar os ^y e ^z, pois com eles será possível encontrar a solução do nosso sistema inicial.



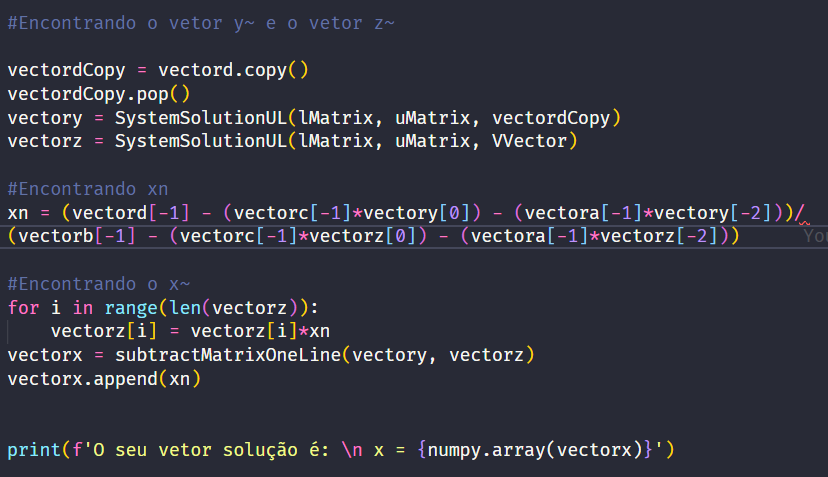


Para encontrarmos os vetores citados anteriormente foi utilizado o algorítmo proposto pelo EP, que foi:



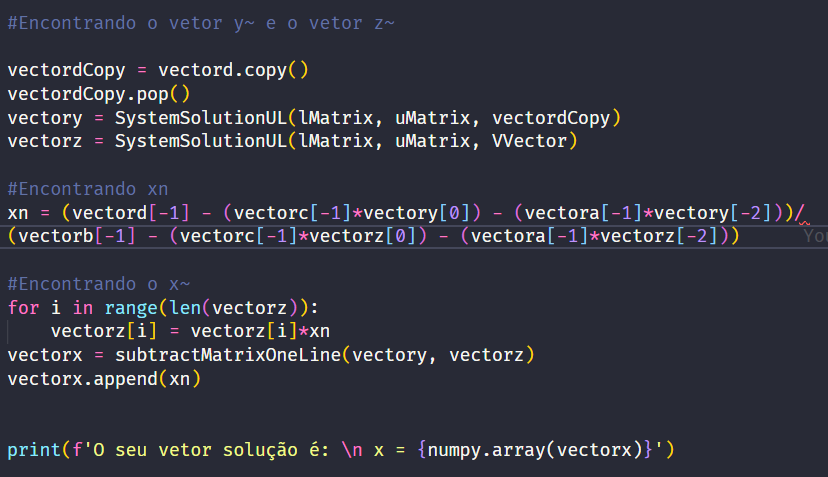
Assim, traduzindo para a linguagem python temos:

Observe que o algorítmo é achamado a partir de uma função, podendo ser utilizado para qualquer sistema tridiagonal.



Após encontrarmos ^y e ^z, basta seguirmos as recomendações de cálculos mostrados no enunciado:

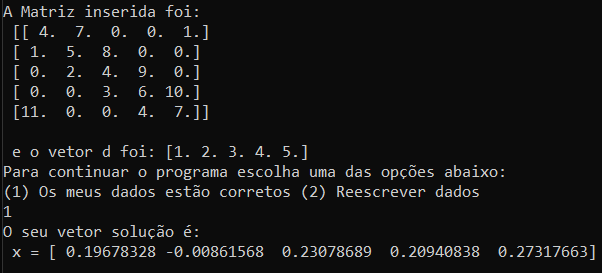


Que, transformando para a linguagem python, obtemos:

Assim, o usuário obterá o vetor solução do sistema inicial.

Segue abaixo o funcionamento da segunda soluçãoTexto

Descrição gerada automaticamente



**Solução 3 – Testes automatizados**

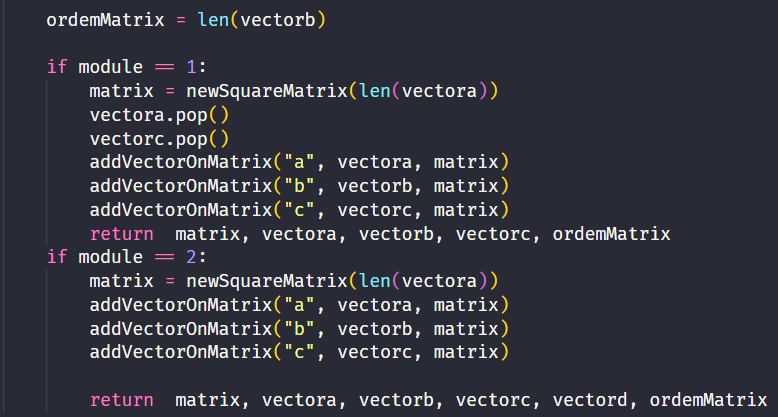
Basicamente na solução 3 o usuário possui a opção de realizar os testes 1 e 2 a partir de vetores a, b, c e d definidos no enunciado do EP, ou seja:



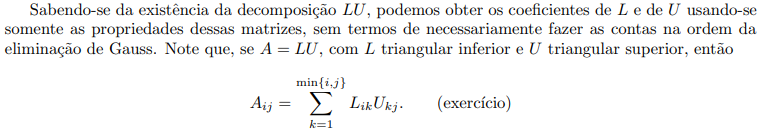


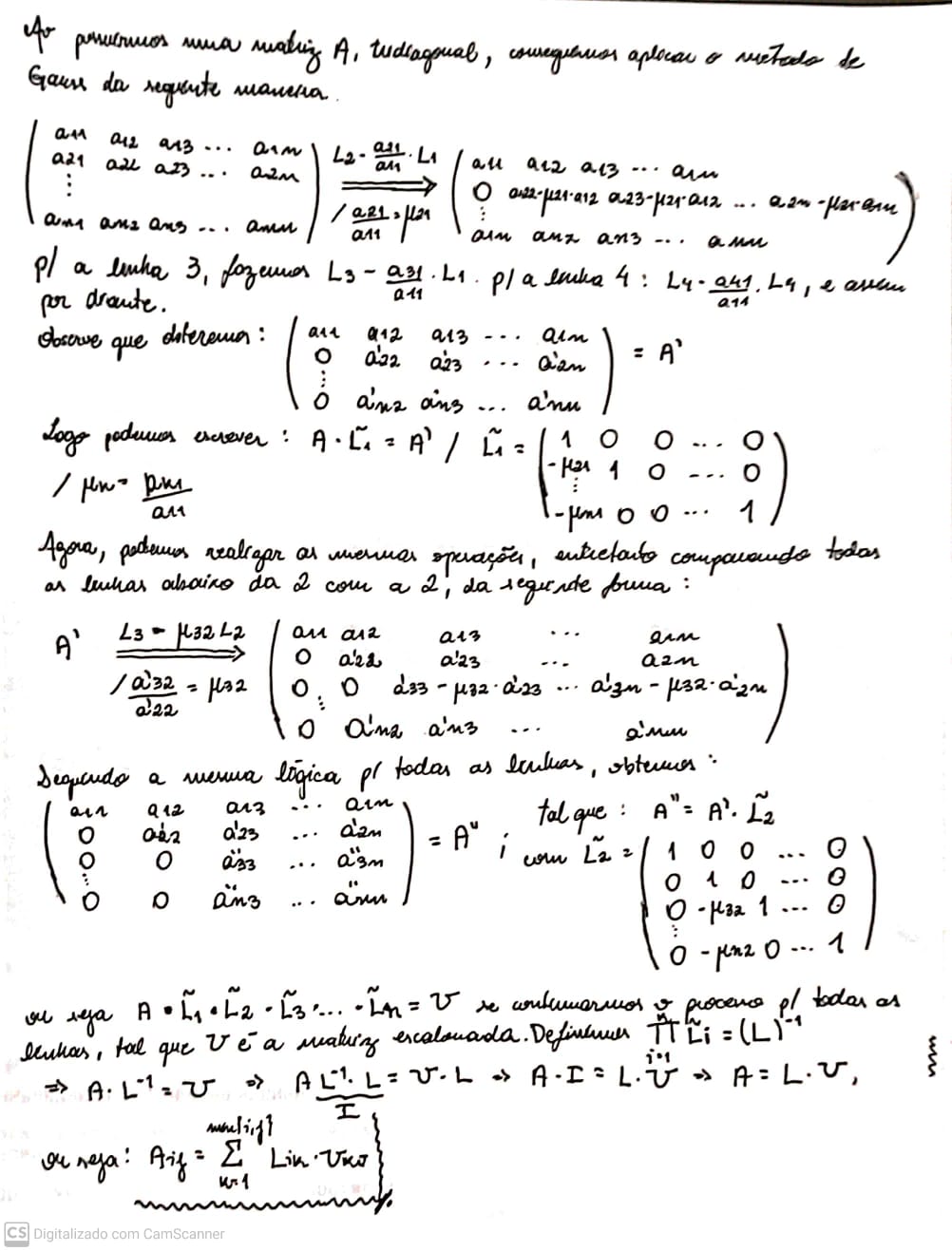
Que, ao passarmos para o algorítmo, oTexto

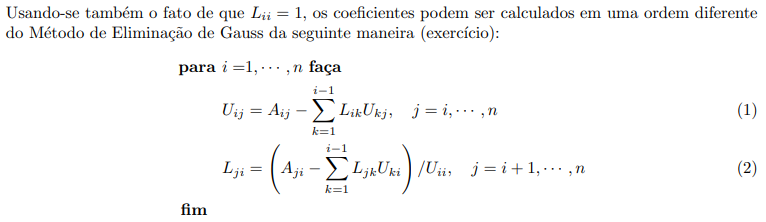
Descrição gerada automaticamentebtemos:

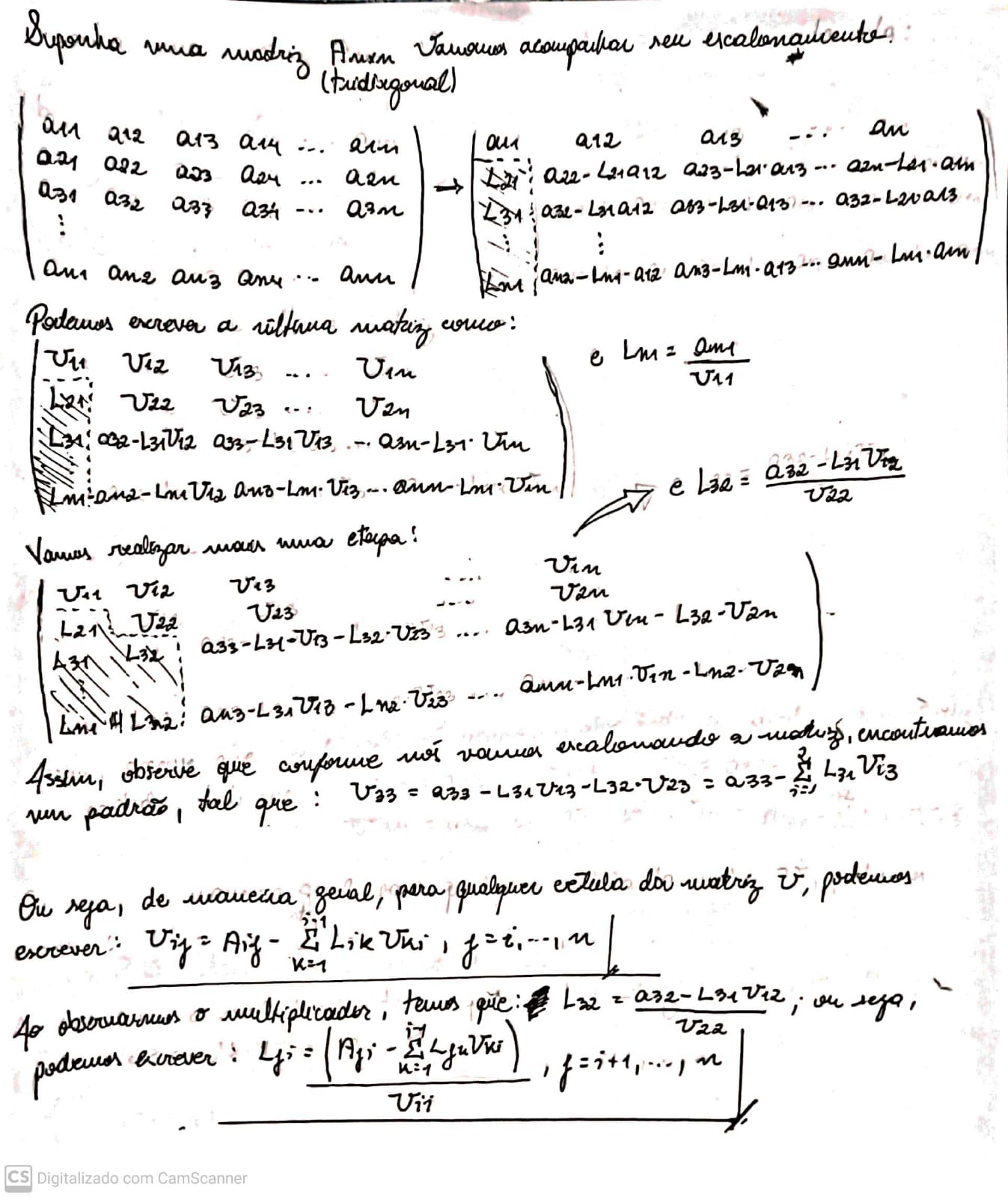


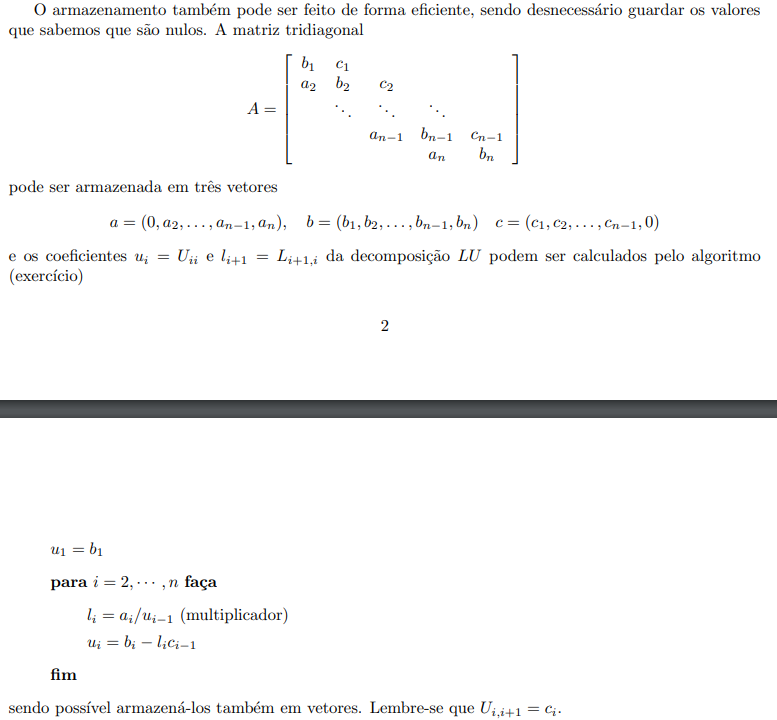
**Exercícios:**

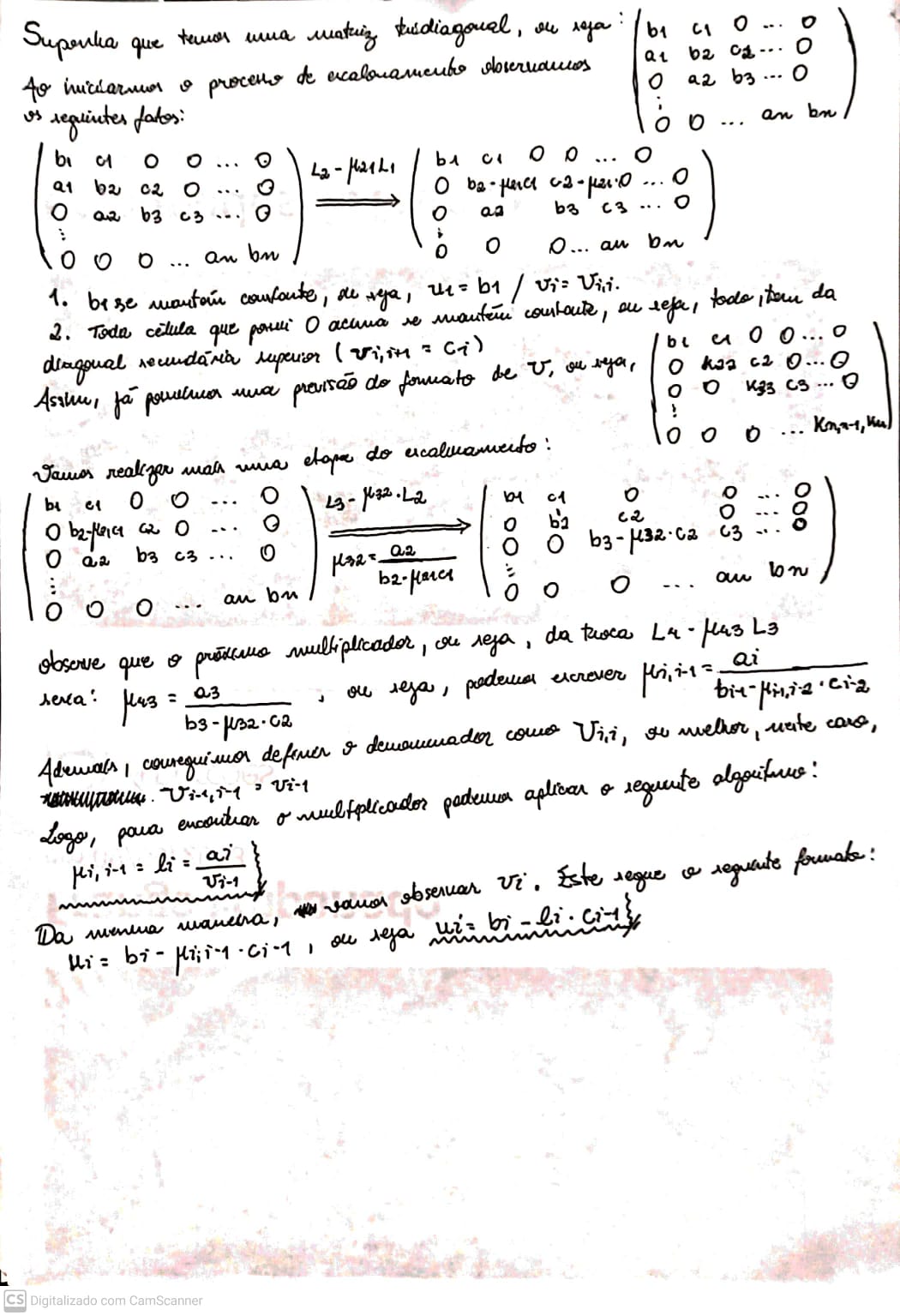
1. ****

****

1. ****

****

1. ****

****