Para este EP foi implementado o algoritmo de householder com pivoteamento das colunas, assim, podendo resolver sistemas lineares sobredeterminados Ax = b sendo uma matriz com posto completo ou incompleto.

O método foi implementado primeiramente faz um pré-processamento, encontrando o maior elemento da matriz A, Beta, dada e em seguida divide todos elementos de A por esse elemento. Aí sim começa as iterações de k indo de 0 até n-2. Primeiro procurando colunas de k+1 até n de A que tenham módulo maior que a coluna k para troca-las em caso positivo. Em seguida encontra r, u e gama para calcular a submatriz $A_{k:n,k:m}$, armazenando u abaixo da diagonal principal de A, já que, essas são todas nulas. Agora calcula a submatriz $A_{k:n,k:m}$ $A_{k:n,k:m}$ - $gama_k u_k u_k^t A_{k:n,k:m}$. Ao final de todo o método é a matriz A é multiplicada por Beta para que ela volte a seus elementos a ordem de grandeza inicial.

Eficácia Para evitar overflows achamos o maior elemento da matriz A e dividimos todos os elementos dela por esse elemento, fazendo assim que o maior elemento seja 1 e evitando que ao ser elevado ao quadrado não estourasse. E o término multiplicamos de novo pelo maior elemento do vetor para voltar a resposta correta.

Em teste feitos usando rand() do C++ não teve nenhum overflow e nenhum underflow.

Para tentar evitar os erros atrelados, consideremos que um elemento é numericamente zero, consideramos que se e somente se a norma da maior coluna for menor que ϵ, sendo ϵ a precisão da máquina, consideramos esse elemento como zero.

Eficiência Como os métodos feitos nesse EP foram feitos em C++, foi implementado a versão dos algoritmos por linha, já que, nessa linguagem as matrizes são armazenadas por linhas, deixando assim, o programa mais rápido por não chamar um número maior que o necessário de pages para a memória e usando o conteúdo em memória muito mais eficientemente, já que, na ordem de elementos acessados, sempre estarão um do lado outro, sendo assim muito mais rápido o acesso ao conteúdo desejado.

A versão do algoritmo implementado, é calculada só a multiplicação implícita de QA visto que o calculo explícito é muito mais custoso.

Como guardar informação em memória é muito custoso, não há variáveis que não sejam essenciais para o calculo do método. Ou seja, as contas foram feitas todas sendo guardadas diretamente na[as] variáveis principais, evitando o uso de variáveis auxiliares, deixando o programa um pouco menos legível mas mais eficiente por não gastar tempo guardando em memória desnecessariamente.

Para os testes feitos com sistemas 20x10, 200x100 e 2000x1000 obtivemos as seguintes tempos para sistemas sobredeterminados:

20x10	200x100	2000x1000
0,000316	0.156678	57.8958

IMPORTANTE Para compilar o ep basta dar um "make" e em seguida executá-lo com "./ep3".