Problem 1 (Question 2.13)

(1)

Note that

$$\begin{aligned}(A+uv^T)\left(A^{-1}-\frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}\right) &= I+uv^TA^{-1}-\frac{uv^TA^{-1}+uv^TA^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}\\ &= I+uv^TA^{-1}-\frac{u(1+v^TA^{-1}u)v^TA^{-1}}{1+v^TA^{-1}u}\\ &= I+uv^TA^{-1}-uv^TA^{-1}\\ &= I\end{aligned}$$

The other side can be argued in a similar fashion. This proves that

$$(A+uv^T)^{-1}=A^{-1}-\frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}$$

More generally, we have

$$\begin{aligned}(A+UV^T)(A^{-1}-A^{-1}UT^{-1}V^TA^{-1}) &= I+UV^TA^{-1}-UT^{-1}V^TA^{-1}-UV^TA^{-1}UT^{-1}V^TA^{-1}\\ &= I+UV^TA^{-1}-U\left[T^{-1}+V^TA^{-1}UT^{-1}\right]V^TA^{-1}\\ &= I+UV^TA^{-1}-U\left[T^{-1}(I+V^TA^{-1}U)\right]V^TA^{-1}\\ &= I+UV^TA^{-1}-UV^TA^{-1}\\ &= I\end{aligned}$$

The other side can be argued in a similar fashion. This proves that

$$(A+UV^T)^{-1}=A^{-1}-A^{-1}UT^{-1}V^TA^{-1}$$

Problem 2

(a)

Given $LX = B$, consider solving X using forward substitution.
To count the flops, consider solving each columns of x independently

$$\begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ \vdots \\ x_{n1} \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ \vdots \\ b_{n1} \end{bmatrix}$$

Note that

$$x_{k1} = \frac{b_{k1} - \sum_{i=1}^{k-1} l_{ki} x_{i1}}{l_{kk}}$$

In the kth level, we need 1 division, $k-1$ multiplication, and $k-1$ additions. Hence, the total number of operation (flops) per column is given by

$$\underbrace{\sum_{i=1}^{n} 1}_{\text{division}} + \underbrace{\sum_{i=1}^{n} (i-1)}_{\text{multiplication}} + \underbrace{\sum_{i=1}^{n} (i-1)}_{\text{addition}} = n^2$$

Since there are n columns, the total number of flops used by forward substitution is

$$F = n^3$$

(b)

Partition L, X, B into sub-matrices of size $b \times b$, we perform forward substitution block-wise

```
for i in range(n/b) do:
    for j in range(n/b) do:
    Read X[i,j] into cache
    Read B[i,j] into cache
        for k in range(i-1) do:
            Read L[i,k], X[k,j] into cache
            B[i,j] -= L[i,k]X[k,j]
        end for
    solve L[i,i]X[i,j] = B[i,j] using forward substitution
    Write X[i,j] back to memory
    end for
end for
```

Counting the total number of words moved between cache and memory, we get

$$\text{For reading } X[i,j] \leq \left(\frac{n}{b}\right)^3 b^2$$

$$\text{For reading } B[i,j] = \left(\frac{n}{b}\right)^2 b^2$$

$$\text{For reading } L[i,j] \leq \left(\frac{n}{b}\right)^3 b^2$$

$$\text{For writing } X[i,j] = \left(\frac{n}{b}\right)^2 b^2$$

Hence, the total number of words moved W is given by

$$W = \frac{2n^3}{b} + 2n^2 = O(\frac{n^3}{\sqrt{M}})$$

Which is done by taking $b \approx \frac{\sqrt{M}}{3}$. Note that this bound is the same bound as in matrix multiplication.

(c)

Using results from HW 1.10, we see that for each k,

$$(1-\delta_k)L[i,k]X[k,j] \leq \text{fl}(L[i,k]X[k,j]) \leq (1+\delta_k)L[i,k]X[k,j]$$

As a result, we can bound

$$\begin{aligned}
\text{fl}(\hat{B}[i,j]) &= \text{fl}(B[i,j] - \sum_{k=1}^{i-1} L[i,k]X[k,j]) \\
&\leq (1+\delta)(B[i,j] - \sum_{k=1}^{i-1} L[i,k]X[k,j]) \\
&= \hat{B}[i,j] + \delta\hat{B}[i,j]
\end{aligned}$$

Using results from HW 1.11, we see that for each sub-matrix $X[i,j]$ solving

$$L[i,i]X[i,j] = \hat{B}[i,j]$$

Gives a solution that satisfies the perturbed problem

$$(L+\delta L)X = (B+\delta)$$

Since X is the right solution for a slightly wrong problem, it follows that the algorithm is backward stable as in HW 1.11.

(d)

```
def recursiveSolver(L, B):
    # Helper function: parition partitions a matrix M into 4 equal pieces
    if size(L) == size(B) < M do:
        solve LX=B by forward substitution
        return X
    else do:
        L_11, L_12, L21, L22 = partition(L)
        B_11, B_12, B21, B22 = partition(L)
        X_11 = recursiveSolver(L_11, B_11)
        X_12 = recursiveSolver(L_11, B_12)
        X_21 = recursiveSolver(L_22, B_21 - L_21 * X_11)
        X_22 = recursiveSolver(L_22, B_22 - L_21 * X_12)
        return [[X_11, X_12], [X_21, X_22]]
```

Let W denote the number of words moved, then we have the recurrence

$$W(n) = 4T(\frac{n}{2}) + 3 \cdot 4\left(\frac{n}{2}\right)^2$$

The recurrence stops when the matrices fit in the cache. The base case $W(b) = 3b^2$. Solving this recurrence gives us

$$W = O(\frac{n^3}{\sqrt{M}})$$

Which is still the same as matrix multiplication.

(e)

By problem 1.11, we see that X_{11} and X_{21} is the solution of

$$(L + \delta L)X = B$$

Since X_{11}, X_{21} are perturbed, it follows that when solving $LX = B$ for X_{21} and X_{22}, the matrix B will also be perturbed. Hence, by the same argument as part (c), we see that the final solution X is given by

$$(L + \delta L)X = B + \delta B$$

Which is still backward stable.