Problem 1

Fix $x, y \in \mathbb{R}$. Consider $\tilde{y} = \text{fl}(y) \in \mathbb{R}$, we proceed to find $\tilde{x} \in \mathbb{R}$ for backward stability, *i.e.*,

$$\text{fl}(\text{fl}(x) \div \text{fl}(y)) = \tilde{x} \div \tilde{y} = \tilde{x} \div \text{fl}(y)$$

Let $\tilde{x} = \text{fl}(\text{fl}(x) \div \text{fl}(y))\text{fl}(y)$; it remains to show that $\frac{|x-\tilde{x}|}{|x|} \leq \epsilon$ for some $\epsilon > 0$. Assuming $\text{fl}(y) \neq 0$,

$$\begin{aligned}
\epsilon_{machine} &\geq \frac{|\text{fl}(\text{fl}(x) \div \text{fl}(y)) - \text{fl}(x) \div \text{fl}(y)|}{|\text{fl}(x) \div \text{fl}(y)|} \\
&= \frac{|\text{fl}(\text{fl}(x) \div \text{fl}(y))\text{fl}(y) - \text{fl}(x)|}{|\text{fl}(x)|} \\
&= \frac{|\tilde{x} - \text{fl}(x)|}{|\text{fl}(x)|}
\end{aligned}$$

Now since $\dfrac{|x - \text{fl}(x)|}{|x|} \leq \epsilon_{machine}$,

$$\begin{aligned}
\frac{|x - \tilde{x}|}{|x|} &\leq \frac{|x - \text{fl}(x)|}{|x|} + \frac{|\text{fl}(x) - \tilde{x}|}{|x|} \\
&\leq \epsilon_{machine} + \epsilon_{machine}\frac{|\text{fl}(x)|}{|x|} \\
&\leq \epsilon_{machine} + \epsilon_{machine}\frac{|x| + \epsilon_{machine}|x|}{|x|} \\
&\leq \epsilon_{machine}\,(2 + \epsilon_{machine})
\end{aligned}$$

□

Problem 2

All single-precision floating point numbers has the form

$$x = (-1)^s 2^E (1 + f)$$

where $s = 0, 1$ determines the sign of x, E ranges from -127 to 128, and $f = \sum_{j=1}^{23} f_j 2^{-j}$, or $f = 0.f_1 f_2 f_3 \cdots f_{23}$ in binary representation. (f_j's are 0 or 1.) Surely we see if $s = 0$, $E = 128$, and all f_j's are all 1, then the largest possible single-precision floating point number

$$\begin{aligned} x_{max} &= (-1)^0 2^{128} \left(1 + \sum_{j=1}^{23} 2^{-j}\right) \\ &= \sum_{j=0}^{23} 2^{128-j} = \sum_{k=105}^{128} 2^k \end{aligned}$$

is a sum of positive integers therefore itself is an integer; however, this does not capture the "holes" of the floating point number system. Observe for $E \geq 23$,

$$\begin{aligned} x &= (-1)^0 2^E \left(1 + \sum_{j=1}^{23} f_j 2^{-j}\right) \\ &= 2^E + \sum_{j=1}^{23} f_j 2^{E-j} \end{aligned}$$

is not much but sum of integers. Consider

$$\begin{aligned} z &= (-1)^0 2^{24} \left(1 + \sum_{j=1}^{24} 2^{-j}\right) \\ &= \sum_{k=0}^{24} 2^k \\ &= \underbrace{111 \cdots 1}_{\text{(25-digits)}} \end{aligned}$$

This number can't be represented in single-precision floating point number system since it has more nontrivial digits than 23. □

Problem 3, 4

Solutions to both problem 3 and 4 are attached in the codes below.

```
function hw1
%% build sample matrix A
n = 10;
A = zeros(n);
A(1,1) = 2;
A(1,2) = -1;
for i=2:n-1
    A(i,i-1) = -1;
    A(i,i) = 2;
    A(i,i+1) = -1;
end
A(n,n-1) = -1;
A(n,n) = 2;

%% QR
[Q, R] = myhouseholderQR(A);
resQR = A - Q*R;
format shortEng;
fprintf('Residual of the Householder QR = \n');
disp(norm(resQR(:)));

%% LU
[L,U] = myLU(A);
resLU = A - L*U;
format shortEng;
fprintf('Residual of LU = \n');
disp(norm(resLU(:)));
end

function [Q, R] = myhouseholderQR(A)
% [Q, R] = householderQR(A)
% Computes the QR-decomposition of input matrix A
[m,n] = size(A);
Q = eye(m);
for j=1:n-1
    x = A(j:m, j);
    v = [zeros(j-1, 1); x + norm(x) * eye(m-j+1,1)];
    Pv = eye(m) - (2*v)*v'/(v'*v);
    A = Pv * A;
    Q = Pv * Q;
```

```
end
Q = Q';
R = A;
end

function [L, U] = myLU(A)
[m,n] = size(A);
assert(m==n, 'Input is not a square matrix.\n');
clear m;

L = eye(n);
U = A;
for j=1:n-1
    for i=j+1:n
        factor = U(i,j) / U(j,j);
        U(i,:) = U(i,:) - factor * U(j,:);
        L(i,:) = L(i,:) - factor * L(j,:);
    end
end
L = inv(L);
end
```