QUESTION – 1

Since Newton's method will be in use, we are going to need the derivative of the given $f(x)$

1-a) $f(x) = 2cosh(x/4) - x$

$f'(x) = \frac{sinh(x/4)}{2} - 1$

2-b) $f(x) = \begin{cases} \frac{sin(x)}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases}$

$f'(x) = \begin{cases} \frac{cos(x)x - sin(x)}{x^2}, & x \neq 0 \\ 0, & x = 0 \end{cases}$

If the limits around x = 0 are checked it can be seen that function is continuous.

```
function hw2q1()

% for part 1a
% returnsvalue of f(x)
f = @(x) 2*cosh(x/4) - x;

% returns the value of f'(x)
fd = @(x) (sinh(x/4))/2 - 1;

% for part 1b
    function y = f2e(x)
        if(x ~= 0)
            y = sin(x) / x;
        else
            y = 1;
        end
    end

    function y = f2de(x)
        if (x ~= 0)
            y = (cos(x)*x - 
sin(x))/x^2;
        else
            y = 0;
        end
    end

f2 = @(x) f2e(x);
f2d = @(x) f2de(x);

    % Function that checks for every 
interval
    function findroots(f, fd, a, b, 
nprobe, tol)
        delta = (b - a) / nprobe;
        for n = a:delta:b
            % If the function changes 
sign in given interval
            % newton function is 
called
            if ((f(n) * f(n+delta)) < 
0)
                disp(newton(n, 
n+delta, f, fd, tol)); % Finding roots 
and
                % Displaying them.
            end
        end
    end
```

```
    % Newton method that takes input as a,b 
interval function and it's
    % derivative, and a tolerance value.
    function x = newton(a, b, f, fd, tol)
        xk = a - f(a)/fd(a);
        xk0 = a;
        while(abs(xk - xk0) >= tol*(1 + 
abs(xk)) || abs(f(xk)) >= tol)
            xk0 = xk;
            xk = xk0 - f(xk0)/fd(xk0);

            if( abs(f(xk)) >= 
0.5*abs(f(xk0)))
                % If this statement is 
satisfied we have to reduce the
                % interval. This is done by 
applyning bisection method to
                % the given interval over 3 
times. Function returns the new
                % interval that newton 
method will be searching on.
                [a, b] = bisection(a, b, 
f);
                xk = a;
            end

        end
        x = xk;
    end

    function [a, b] = bisection(a, b, f)
        for i = 0:1:2
            xk = (b-a) / 2;
            if (f(xk) * f(a) <= 0)
                b = xk;
            elseif (f(xk) * f(b) < 0)
                a = xk;
            end
        end
    end
findroots(f, fd, 0, 10, 10, 10^-7);
findroots(f2, f2d, -10, 10, 20, 10^-7)
end
```

Roots of 1-a: 2.357551053877402, 8.507199570713095

Results of 1-b: -9.424777960769380, -6.283185307179586, -3.141592653589793, 3.141592653589793, 6.283185307179586, 9.424777960769380

QUESTION – 2

a-) Since taking derivative of this function is super easy, newton method can be applied here. Bisection and secant methods would have worked. Newton method is superior over bisection because of the speed of convergence, and it will preferred over secant since you have the real value of derivative rather than an approximation.

b-) It is clear fact that this function's derivative does not point to the root all the time. It changes a lot by a large margin. This oscillation will slow down newton and secant methods. It can even create disconvergence. Bisection is method should be used in order to find roots for such functions.

c-) This function is clearly not suitable for Newton method. Since we cannot explicitly define $f'(x)$. Bisection method can be used. However secant method will be choice. We don't need the derivative of the function in secant method, it can be approximated. Which'll grant us greater convergence over bisection method.

QUESTION – 3

```
function hw2q2()
    function x = solve (n, lower_d, d, upper_d, b)

        % Gaussian elemination
        upper_d(n) = 0;
        for i = 1:n-1
            k = lower_d(i)/d(i);
            d(i+1) = d(i+1) - upper_d(i)*k;
            b(i+1) = b(i+1) - b(i)*k;
            lower_d(i) = 0;
        end

        % Finding x[n];
        x(n) = b(n)/d(n);

        % Backward substitution
        for i = n-1:-1:1
            x(i) = (b(i) - x(i+1) * upper_d(i))/d(i);
        end
    end

    % Inputs
    n = 10;
    lower_d = [-1;-2;-3;-4;-5;-6;-7;-8;-9];
    d = [3;6;9;12;15;18;21;24;27;30];
    upper_d = [-2;-3;-4;-5;-6;-7;-8;-9;-10];
    % Chosen b
    b = [1;2;3;4;5;6;7;8;9;10];
    % Displaying result
    x = (solve(n, lower_d, d, upper_d, b));
    disp(x);
end
```

Chosen b = [1, 2, 3, 4, 5, 6, 7, 8, 9 ,10]

Results: X = [0.999378917057196 0.999068375585794 0.998343778819190 0.996739314550280 0.993168087629157 0.985094009372706 0.966550247223275 0.923373891931568 0.821680186199412 0.579837389193157]

QUESTION – 4

a-)

$$A = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 \end{bmatrix}$$

For the given A matrix we need to swap the 3rd and 4th rows. This can be achive with the permutation matrix P.

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Since PA is an upper triangular matrix (U), lower triangular matrix(L) will be an identity matrix.

$$PA = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = U = IU \qquad L = I$$

b-)

x can be calculated with the help of LU decomposition.

$$\begin{aligned} Ax &= b \\ A &= LU \\ L(Ux) &= b \\ Ly &= b \\ Ux &= y \end{aligned}$$

Since L is an identity matrix it can be said that $y = b$ thus solution of $Ux = b$ is required which is done by backward substitution.

$$Ux = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 26 \\ 9 \\ 1 \\ -3 \end{bmatrix}$$

$$x_4 = b_4 = -3$$

$$x_3 = -b_3 - 2x_4 = -1 + (-2 * -3) = 5$$

$$x_2 = \frac{b_2 - 3x_2 - 2x_4}{4} = \frac{9 - 3 * 5 - 2 * -3}{4} = 0$$

$$x_1 = \frac{26 - 6x_2 - 7x_3 - 8x_4}{5} = \frac{26 - 6 * 0 - (7 * 5) - (2 * -3)}{5} = 3$$