```
% Index number = 180631J
% Therefore the required parameters
A = 6;
B = 3;
C = 1;
```

Prescribed Filter specifications

```
tilde_A_p = 0.03+0.01*A;    % Maximum passband ripple
tilde_A_a = 45 + B;         % Minimum stopband attenuation
omega_p1 = C*100 + 300;     % Lower passband edge
omega_p2 = C*100 + 700;     % Upper passband edge
omega_a1 = C*100 + 150;     % Lower stopband edge
omega_a2 = C*100 + 800;     % Upper stopband edge
omega_s  = 2*(C*100 +1200); % Sampling frequency
```

Derivation of filter Parameters

```
B_t1 = omega_p1 - omega_a1; % Lower transition width
B_t2 = omega_a2 - omega_p2; % Upper transition width
B_t  = min(B_t1,B_t2);      % Critical transition width
omega_c1 = omega_p1-B_t/2;  % Lower cutoff frequency
omega_c2 = omega_p2+B_t/2;  % Upper cutoff frequency
T = 2*pi /omega_s;          % Sampling period
```

Derivation of the Kaiser Window Parameters

```
tilde_delta_p  =  (10^(0.05*tilde_A_p) -1)/(10^(0.05*tilde_A_p)  +1);
tilde_delta_a  = 10^(-0.05*tilde_A_a);
delta = min(tilde_delta_p, tilde_delta_a);

A_a = -20*log10(delta);% Actual stopband attenuation

% Choose parameter alpha as,
if A_a <=21
    alpha = 0;
elseif 21 < A_a && A_a <=50
    alpha = 0.5842*(A_a - 21)^0.4 + 0.07886*(A_a - 21);
else
    alpha = 0.1102*(A_a - 8.7);
end

% Choose parameter D as,
if A_a <= 21
    D = 0.9222;
else
    D = (A_a - 7.95)/14.36;
end

% Select the lowest odd value of N that satisfies the inequality
N = ceil(omega_s*D/B_t + 1);
```

```
if mod(N,2) ==0
    N = N+1; % If calculated N is evn, make it odd by adding 1
end
```

Creating the Kaise Window

```
n = -(N-1)/2:1:(N-1)/2;                               % Range of the Kaizer window
beta = alpha*sqrt(1-(2*n/(N-1)).^2);                  % betas for I(beta)
terms = 50;                                           % Number of terms to be considered for bessel
w_k_nT = ZerothOrderModifiedBessel(beta,terms)...
            /ZerothOrderModifiedBessel(alpha,terms);    % Calculating window coefficients
stem(n,w_k_nT,'filled');
title('Kaiser Window - Time Domain');
xlabel('Samples(n)');
ylabel('Amplitude');
```

Derivation of The Ideal Impulse Response

```
h_d_nT = (sin(omega_c2*n*T) - sin(omega_c1*n*T))./(pi*n); % For each n != 0
h_d_nT((N+1)/2) = (omega_c2 - omega_c1)*(2/omega_s);      % For n = 0
stem(n,h_d_nT,'filled');
title('The Ideal Impulse Response');
xlabel('Samples(n)');
ylabel('Amplitude');
```

Truncating the Ideal Impulse Response to obtain Finite Impulse Response(Anti-Causal)

```
h_nT = h_d_nT.*w_k_nT; % Windowing using Kaiser window
stem(n,h_nT,'filled');
title('Finite Impulse Response- Anti-Causal')
xlabel('Samples(n)');
ylabel('Amplitude');
```

Finite Impulse Response(Causal)

```
n_causal = 0:1:N-1; % Making the range of n positive
stem(n_causal,h_nT,'filled');
title('Finite Impulse Response-Causal')
xlabel('Samples(n)');
ylabel('Amplitude');
grid on;
```

Plotting the magnitude response of filter in the range (0,omega_s/2)

```
%fvtool(h_nT,'magnitude')
[H_ejomegaT, omega] = freqz(h_nT);
omega = (omega/pi)*(omega_s/2); % rad/s = (normalized freq)*(sampling freq/2)
magnitude = 20*log10(abs(H_ejomegaT));
plot(omega, magnitude);
```

```
title('Magnitude Response of Filter');
xlabel('Angular Frequency (rad/s)');
ylabel('Magnitude (dB)');
grid on;
```

Magnitude response of the digital filter for the frequencies in the passband

```
plot(omega, magnitude);
xlim([omega_p1 omega_p2]);
title('Magnitude Response of Filter  in the passband');
xlabel('Angular Frequency (rad/s)');
ylabel('Magnitude (dB)');
grid on;
```

Local Function definitions

```
function value = ZerothOrderModifiedBessel(x,terms)
value = 1;
for k = 1:terms
    value = value + ((1/factorial(k))*(x/2).^k).^2;
end
end
```