Appendix 2: Program code

Abel-Smith Algorithm

main_program

```
clear all
close all
clc

B =56;% GHz
lambda_0 = 1550; %nm
D=16; %ps/nm/km
L = 23; %km
c = 299792458; %m/s
alpha = lambda_0^2*B^2*D*L/(4*pi*c*1000);
N_IIR = ceil(lambda_0^2*B^2*D*L/(2*c*1000));
[rho,theta] = abel_smith(alpha,N_IIR);
```

abel_smith function

```
function [ rho,theta ] = abel_smith( alpha,N_IIR )
% The filter coefficients for cascaded filter
% rho: radius
% Theta: phase
% xi: facultative correction factor between 0.75 and 0.85

xi = 0.8;
omega_seg = abel_smith_divide(alpha, N_IIR);

for i=1:N_IIR
    theta(i) = 0.5 * (omega_seg(i) + omega_seg(i+1));
    delta(i) = 0.5 * (omega_seg(i+1) - omega_seg(i));
    miu(i) = (1 - xi*cos(delta(i)))/(1-xi);
    rho(i) = miu(i) - sqrt(power(miu(i),2)-1);
end
```

abel_smith_divide

```
function [ omega_seg ] = abel_smith_divide( alpha,N_IIR )
% Functions calculates the segmentation for the abel smith algorithm for a
% given alpha where alpha= lambda_o^2*B^2*D*L/(4*pi*c)
```

```
beta = ceil(2*alpha*pi);

omega_seg(1) = -pi;
omega_seg(N_IIR+1) = pi;

for i=2:N_IIR
   omega11 = (2*beta + sqrt(4*beta^2 - 8*alpha*(4*pi - 2*alpha*(omega_seg(i-1))^2
+2*beta*omega_seg(i-1))))/(4*alpha);
   omega12 = (2*beta - sqrt(4*beta^2 - 8*alpha*(4*pi - 2*alpha*(omega_seg(i-1))^2
+2*beta*omega_seg(i-1))))/(4*alpha);
   if abs(omega11)<abs(omega_seg(i-1))
      omega_seg(i) = omega11;
   else
      omega_seg(i) = omega12;
   end
end

end
```

Bit Error Rate

main_program

```
function main_program()

clc

%% Operating wavelength and related constants
lambda=1550*1e-9;                    % Operating wavelength in meters
c=3e8;                               % Speed of light in m/s
D=16*1e-12/(1e-9*1e3);               % Dispersion in ps/(nm*km)

%% Configuration Parameters For the CD Channel
L=23*1e3;                            % Length of fiber in meters;
Bw=56e9;                             % Bandwidth of the signal in Hz

%% Sampling frequency
B=Bw;

%% Calculating alpha
alpha=lambda^2*B^2*D*L/(4*pi*c);     % Multiplying the above parameters

%% Number of frequency points needed to generate channel impulse response
freq_pts=1024;
```

```
%% Task 1: (For students) Write a function which generates the impulse response of the CD
channel
%% Generating channel impulse response
h_CD=impulse_response_channel(alpha,freq_pts);

%% All-pass filter coefficients after optimization
load rho, load thet, load phi

%% Task 2: (For students) Write a function which convolves any input with all-pass equalizer
%% In this case, input is the impulse response of CD channel
GH=conv_anyinput_allpass_eqaulaizer(rho,thet,phi,h_CD);
[val index]=max(abs(GH));

%% Plot GH, ideally it should be an impulse
% stem(abs(GH))

%% BER
bit_error_rate(h_CD,index,rho,thet,phi)

end
```

function impulse_response_channel

```
function h_tot=impulse_response_channel(alpha,freq_pts)
%% Generate the impulse response in Time domain
% alpha: channel characteristic alpha= lambda_o^2*B^2*D*L/(4*pi*c)
% freq_pts: Number of Frequency points
% h_CD: column vector with time domain values


%% Sampling Frequency Response
omega = -pi:2*pi/freq_pts:pi-2*pi/freq_pts;
H_CD = exp(-1i*alpha*omega.^2);
H_CD = ifftshift(H_CD);

%% IFFT and normalization

h_CD = ifft(H_CD);
h_CD = h_CD(:); %assure that it is a column vector
h_tot = fftshift(h_CD);

% return column vector
```

```
stem(abs(h_tot));
end
```

function conv_any_input_alpass_equalizer

```
function output=conv_anyinput_allpass_eqaulaizer(rho,theta,phi,input)
%% Convolution of any Filter specified by rho/theta/ Phi with input
% rho: column vector with N_IIR filter radii
% theta: column vector with N_IIR filter phases
% phi: Phase correction term phi_0
% input: signal to be filtered
% output:filtered signal
% Hint: Use filter() function

input = input(:);
for k=1:length(rho)
   b = [-rho(k)*exp(-1i*theta(k)), 1];
   a = [1, -rho(k)*exp(1i*theta(k))];
   output = filter(b,a,input);
   input = output;
end

output = exp(-1i*phi).*output;
stem(abs(output));
end
```