## Appendix 3 – Code Excerpts

### Matlab Code

The code in MATLAB is divided up into individual sections, allowing for modular changes to be made.

```
%% TIME CONSTANTS %%%%%%%%%%%%%%%%%%%%
dur = 5;                              %duration of sample (used
                                      % for calculation of Ns)
fac = 4;                              %over sampling factor
Fs = fac*44100;                       %sample rate
dt = 1/Fs;                            %sampling interval
Ns = floor(dur*Fs);                   %number of samples
t = dt*(0:Ns-1);                      %length of time in (s)
f = 220;                              %input frequency
c = (2/dt);                           %used in bilinear transform
rds= Fs*pi;                           %conversion from Hz to rad/s
                                      %      (Fs/2) * (2*pi)


%% INPUT %%%%%%%%%%%%%%%%%%%%
xx = 0.1*sin(2*pi*f*t); %%100mV sine wave input of frequency f

%%xx = 0.1*chirp(t,10,dur,20000,'logarithmic');
%% sine sweep input from 10-20k Hz

D = 1;                                %%SET DIST
T = 0.85;                             %%SET TONE

%% OP-AMP GAIN STAGE %%
Rt = D.*100000;                 %% Rt %%%%%%%%%
Rb = ((1-D).*100000)+4700;      %% Rb %%%%%%%%%
Cz = 1*(10^-6);                %% Cz %%%%%%%%%
Cc = 250*(10^-12);             %% Cc %%%%%%%%%

alpha = (1/(Rt*Cc));
beta  = (1/(Rb*Cz));
gamma = (1/(Rb*Cc));

g = alpha+beta+gamma;
v = alpha*beta;
k = alpha+beta;
r = (4+2*g*dt);

b2 = 1;     a2 = 1;        %%maps coefficients to H(s)
b1 = g;     a1 = k;        %%
b0 = v;     a0 = v;        %%
```

```
B0 = b0 + b1*c + b2*(c^2);    %%'''|'''''''''''''''''
B1 = 2*b0 - 2*b2*(c^2);       %%   |
B2 = b0 - b1*c + b2*(c^2);    %%Discretisation by bilinear
A0 = a0 + a1*c + a2*(c^2);    %% transform
A1 = 2*a0 - 2*a2*(c^2);       %%   |
A2 = a0 - a1*c + a2*(c^2);    %%,,,|,,,,,,,,,,,,,,,

B = [B0 B1 B2];           %%maps coefficients to numerator
A = [A0 A1 A2];           %%maps coefficients to denominator


opamp = filter(B,A,xx);    %%output from stage when input is
                           %% filtered through H(z) = B/A

varlist =
{'AO','A1','A2','B0','B1','B2','aO','a1','a2','b0','b1','b2'};
clear(varlist{:}) %% clears variables for use later


%% SIMPLIFIED SIMULATION - DIODE CLIPPER STAGE %%

        R = 2.2e3;        %%Circuit resistance
        is = 2.52e-9;     %%Reverse saturation current of diode
        Vt = 45.3e-3;     %%Diode thermal voltage
        Vo = -0.6:0.01:0.6;  %%Vo for mapping
        Vi = Vo + 2*is*R*sinh(Vo/Vt);  %%Calculation
        x = Vi;
        y = Vo;


diode = interp1(x,y,opamp,'pchip');  %%Extracting all points
from op-amp that lie on the saturation curve where output from
stage = "diode"


%% FULL SIMULATION - DIODE CLIPPER STAGE %%

Vis = opamp;       %%input = output from op-amp
ss = 3*eps;
Vi1 = 0;          %%initialise variables
Vo1 = 0;          %%initialise variables
```

```
%% DIODE SIMULATION LOOP %%%%%
    varlist = {'alpha','beta'};
    clear(varlist{:})
    R = 2.2e3;  %%component values from circuit diagram
    C = 10e-9;
    is = 2.52e-9;
    Vt = 45.3e-3;
    a = 1/(2*R*C);
    b = is/C;
    alpha = a*dt;
    beta = b*dt;

    maxiter = 100;    %%loop will not run for more than 100

    for n=1:Ns
    % read in
    Vi0 = Vis(n);

    % NR loop
    diff = 1;
    i = 0;
    while diff > eps && i < maxiter
        F = (alpha + 1)*ss + beta*(sinh((Vo1+ss)/Vt) +
sinh((Vo1)/Vt)) - alpha*(Vi0 + Vi1 - 2*Vo1);
        J = (alpha + 1) + (beta/Vt)*cosh((Vo1+ss)/Vt);

        ssnew = ss - F/J;
        diff = abs(ss-ssnew);
        ss = ssnew;
        i = i + 1;
    end

    % update
    Vo0 = Vo1 + ss;

    % monitor
    Vos(n) = Vo0;

    % memorize
    Vo1 = Vo0;
    Vi1 = Vi0;
End
%% DIODE OUTPUT

diode = Vos;  %% output from stage = "diode"
```

```
%% TONE STAGE %%
clf;

R4 = 2.2e3; %%component values from circuit diagram
R3 = 6.8e3;
R5 = 6.8e3;
C2 = 0.022e-6;
C1 = 0.1e-6;
Rpot = 20e3; %%assign 20k *(1-T) or T for potential divider
k1 = (1-T)*Rpot;
k2 = T*Rpot;

%%transfer function coefficients
a0 = R3 + R4 + Rpot;
a1 = (C1*R5*(R3+Rpot)+C2*(R3*(R4+R5+Rpot)+R4*(R5+Rpot)));
a2 = C1*C2*R5*(R3*(R4+Rpot)+R4*Rpot);
b0 = (1-T)*Rpot + R3;
b1 = C2*((1-T)*R4*Rpot+R3*(R4+R5+Rpot));
b2 = T*C1*C2*R3*R5*Rpot;

 %%bilinear transform
        B0 = b0 + b1*c + b2*(c^2);
        B1 = 2*b0 - 2*b2*(c^2);
        B2 = b0 -  b1*c + b2*(c^2);
        A0 = a0 + a1*c + a2*(c^2);
        A1 = 2*a0 - 2*a2*(c^2);
        A2 = a0 - a1*c + a2*(c^2);

B = [B0 B1 B2];           %%H(z) Numerator %%
A = [A0 A1 A2];           %%H(z) Denominator %%

tonal = filter(B,A,diode);  %%output from stage when input is
                            %% filtered through H(z) = B/A

%% Final output from circuit = tonal

%% PLOTTING
```

**%% Various Matlab coding for plotting time domain and frequency domain analysis such as “plot()” and “surf()”**

**This is not included as it was case by case basis and differed for each test%%**