A.3 Matlab Code: Source-Frame Extraction

```
function [sourceFramesUtterance,F0_samples_Utterance] = ...
                    SourceFrames(x,fs,normdL,minf0,maxf0,frame_indicator)
%
%   INPUT:  x  - speech waveform
%           fs - sample frequency of x
%           _OPTIONAL_
%           normdL - normalised length of source-frames (256)
%           minf0 - minimum F0 to set lags for (100)
%           maxf0 - maximum F0 to set lags for (350)
%           frame_indicator - pre computed v/uv frame decisions
%
%   OUTPUT: sourceFramesUtterance - normalised voice source waveform
%                               derivative for each voiced pitch period.
%           F0_samples_Utterance - F0 corresponding to each source-frame
%
% Code uses Voicebox Matlab toolkit:
%       http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
% and for accurate F0 estimation:
%       http://www.tik.ee.ethz.ch/db/public/tik/?db= ...
%       publications&form=report_single_publication&publication_id=3227

if nargin < 5
    minf0 = 100;
    maxf0 = 350;
end
if nargin < 3
    normdL = 256;
end

%% CONSTANTS
disp('Note that several parameters are specified inside SourceFrames()');
frameLengthTime = 0.03;      %seconds
frameIncTime = 0.01;         %seconds
frameLengthSamples = floor(frameLengthTime*fs);
frameIncSamples = floor(frameIncTime*fs);
nFFTpts = 1024;%more=better, at cost of computation time (F0 estimation)
alpha = 0.95;    %pre-emphasis slope
P = round(fs/1000)+3;  % rule of thumb for LP order
h = hanning(frameLengthSamples+P);    %window function for LPC autocorr.
%hammingWindow = hamming(normdL)';
blackmanWindow = blackman(normdL)'; % source-frame normalisation window

%% DIRECTORIES
matlabDir = cd;
if ismac()
    voiceboxDir = '/Users/davidvandyke/coding/Matlab/voicebox';% macbook
elseif isunix()
    voiceboxDir = '/data1/Code/DavidV/Voicebox';   % HCC Server
elseif ispc()
```

```
    voiceboxDir = 'X:\Code\DavidV\Voicebox'; % UC desktop
end

%% CODE
sourceFramesUtterance = [];
if nargout > 1
    F0_samples_Utterance = [];
end

%Split speech signal x into frames
cd(voiceboxDir);
frames = enframe(x,frameLengthSamples,frameIncSamples);
nFrames = size(frames,1);
cd(matlabDir);

%Determine F0 for each frame
F0_PLOT = false;
F0 = detect_F0_contour(x,fs,frameLengthTime,frameIncTime,nFFTpts,...
    minf0,maxf0,F0_PLOT);
frameF0samples = round(fs./F0);

%Determine which frames are voiced/unvoiced
if nargin > 5 % in this case have already determined which frames we want
    frameVoicingBoolean = frame_indicator;
    if numel(frame_indicator) ~= nFrames
        error('Number of frames not matched');
    end
else
    frameVoicingBoolean = VoicingDetector(frames,fs,minf0,maxf0);
end

%Determine averaged speech waveform for GCI and GOI detection
averaged = DynamicAveragedSpeech(x,fs,frameIncSamples,F0);
cd(voiceboxDir);
averagedFrames = enframe(averaged,frameLengthSamples,frameIncSamples);
cd(matlabDir);

%Parse all voiced frames
for frameNo = 2:nFrames-1
    %this avoids out of index problems with xframePlus
    %(unlikely first frame is voiced anyhow)
    if frameVoicingBoolean(frameNo)
        %then is a voiced frame
        xframe = frames(frameNo,:);
        aframe = averagedFrames(frameNo,:);

        % STEPS 1 & 2 - determine GCI and GOI
        %1. determine residual (lpc error signal) over whole frame
        xframePlus = [frames(frameNo-1,end-P+1:end),xframe];
        xresidual = LPC_AutoCorrFrame(xframePlus,h,alpha,P);

        %2. determine GCI and GOI
        [GCIs,GOIs] = ...
```

```
            GlottalInstants(aframe,xresidual,frameF0samples(frameNo));

        %3. closed phase auto covariance LPC + Prosody normalisation
        sourceFrames = ClosedPhaseLPC_Covariance(GCIs,GOIs,...
            frames(frameNo-1:frameNo+1,:),P,alpha,...
            frameF0samples(frameNo),normdL);

        %4. Blackman window source-frames
        if ~isempty(sourceFrames)
            for ii = 1:size(sourceFrames,1)
                sourceFrames(ii,:) = sourceFrames(ii,:).*blackmanWindow;
            end

        %5. Store
            sourceFramesUtterance = [sourceFramesUtterance;sourceFrames];
            if nargout > 1
                F0_samples_Utterance = ...
                    [F0_samples_Utterance;frameF0samples(frameNo)];
            end
        end
    end
end
end  % END OF FUNCTION
```

```
function [residual,lpcCoeffs,gain] = ...
            LPC_AutoCorrFrame(framePlus,h,alpha,P)
%
% INPUT: framePlus: frame with P extra samples from previous frame
%        h: windowing function for autocorrelation LPC
%        alpha: pre-emphasis factor
%        P: LP order
%
% OUTPUT: residual: LP residual signal
%         lpcCoeffs: autocorrelation lp coefficients
%         gain: sqrt of power of linear predictors' error

N = length(framePlus)-P;

%pre-emphasis then window
preEmph = filter([1 -alpha],1,framePlus);
data = preEmph.*h';

%lpc predictor coeffs
[lpcCoeffs,gainSqrd] = lpc(data',P); gain = sqrt(gainSqrd);

%calculate residual from lp
residual = zeros(1,N);   %pre-allocation
for n=1:N
    residual(n) = dot(lpcCoeffs,fliplr(framePlus(n:n+P)));
end
end % END OF FUNCTION
```

```
function [GCIs,GOIs] = ...
    GlottalInstants(averagedFrame,frameResidual,frameF0samples)
%
% INPUT: averagedFrame: frame of averaged (low pass) speech
%          frameResidual: frames autocorr lp residual
%          frameF0samples: frame F0 in # samples
%
% OUTPUT: GCIs / GOIs: closure and opening instants per pp
%
% Based on: Drugman, T. and Dutoit, T., INTERSPEECH 2009
%   Glottal closure and opening instant detection from speech signals,
%

%% CONSTANTS & PREALLOCATION
GCIs = [];
GOIs = [];
DEBUG = false; % boolean - plot gci and goi estimates

%% CODE
%determine extrema of sine type waveform and mid points between these.
points = [];
extrema = abs(averagedFrame(2:end)-averagedFrame(1:end-1));

for ii = 2:length(extrema)-1
    %starting from min
    if extrema(ii) > extrema(max([1,ii-1])) && ...
            extrema(ii) > extrema(min([length(extrema),ii+1]))
        points = [points;ii];
    elseif extrema(ii) < extrema(ii-1) && extrema(ii) < extrema(ii+1)
        points = [points;ii];
    end
end

% REMOVE any points that are very near to each other (due to small
% problem variations in the averaged waveform)
minDIST = round(0.2*frameF0samples);
ii = 1;
while ii < length(points)-1
    if points(ii+1) - points(ii) < minDIST
        points(ii+1) = round(mean((points(ii+1)+points(ii))/2));
        points(ii) = [];
        ii = max([1,ii-1]);
    end
    ii = ii +1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if DEBUG
    figure;
    plot(averagedFrame,'r');
    hold on
    for ii = 1:length(points)
        plot(points(ii),averagedFrame(points(ii)),'kx');
```

```
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%start parsing from first minima of averaged waveform
firstMin = -1;
%should always be in first 4, except when averaged waveform is bad.
for ii = 1:length(points)
    cond1=averagedFrame(max([1,points(ii)-3]))>averagedFrame(points(ii));
    cond2 = averagedFrame(min([...
        length(averagedFrame),points(ii)+3]))>averagedFrame(points(ii));
    if cond1 && cond2
        firstMin = ii;
        break;
    end
end
if firstMin == -1
    % return GCIs and GOIs as []
    return;
elseif firstMin > 1
    points(1:firstMin-1) = [];  %start from first minima
    if length(points) < 5
        return;
    end
end

%use frameResidual - find extrema of within containers.
pointNo = 1;
while pointNo +3<length(points)
    [~,gciIndex] = ...
        max(abs(frameResidual(points(pointNo):points(pointNo+1))));
    GCIs = [GCIs;gciIndex+points(pointNo)];
    [~,goiIndex] = ...
        max(abs(frameResidual(points(pointNo+2):points(pointNo+3))));
    GOIs = [GOIs;goiIndex+points(pointNo+2)];
    pointNo = pointNo + 4;
end
end % END OF FUNCTION
```

```
function [sourceFrames] = ClosedPhaseLPC_Covariance(GCIs,GOIs,...
                                  frames,P,alpha,frameF0samples,normdL)
%
% INPUT: GCIs/GOIs: closure and opening instants
%        frames: 3 x frameLength - middle frame is of interest.
%                   other 2 are just to allow LPC analysis
%        P: linear predictor order
%        alpha: pre-emphasis factor (deal with 6dB/octave slope)
%        frameF0samples: F0 in samples for each frame of frames
%        normdL: source-frame normalisation length
%
% OUTPUT: sourceFrames: collection of prosody normalised derivative
%                       glottal waveform estimates
```

```
%
% ISSUES: deals by cases with closed phases which are small (<2P)

%% CONSTANTS & PREALLOCATION
sourceFrames = [];
npp = length(GCIs); %number pitch periods
if length(GOIs) ~= npp
    warning('Mismatch GCI and GOI numbers...');
    return;
end
Psmall = P-2; %for high pitch voiced periods we may use a
              %smaller predictor order for LPC analysis
MAX_EXTENSION = 8; %extend the goi by at most MAX_EXTENSION-1 samples
                   %to have  2P samples for LPC covariance analysis
frameLengthSamples = length(frames(2,:));
three_frames = [frames(1,:),frames(2,:),frames(3,:)];
preEmph = filter([1 -alpha],1,frames(2,:)); %pre emphasise speech

%% CODE
for ii = 1:npp
    % Do closed phase LP
    usedP = P;
    %CASE 1: all good
    if GOIs(ii) - GCIs(ii) > 2*P
        lpcCov = arcov(preEmph(GCIs(ii):GOIs(ii)),P);
    %CASE 2: try using a slightly smaller prediction order
    elseif GOIs(ii) - GCIs(ii) > 2*Psmall
        lpcCov = arcov(preEmph(GCIs(ii):GOIs(ii)),Psmall);
        usedP = Psmall;
    else
        extend_by = 2*P - (GOIs(ii)-GCIs(ii));
        %CASE 3: extend the closed phase - by lengthing the opening
        %instant (which is harder to detect - and including has less
        %effect on error as opening is gradual)
        if extend_by < MAX_EXTENSION && GOIs(ii)+extend_by <= ...
                frameLengthSamples
            speech = preEmph(GCIs(ii):GOIs(ii)+extend_by);
            lpcCov = arcov(speech,P);
        else
            %CASE 4: skip this pitch period
            lpcCov = [];
        end
    end

    % Now use lpcCov and determine residual
    if ~isempty(lpcCov)
        % absolute to relative indices:
        residualLength = 3*frameLengthSamples-usedP;
        relativeIndexLeft = frameLengthSamples-usedP-frameF0samples;
        relativeIndexRight = frameLengthSamples-usedP+frameF0samples;
        residual = zeros(1,residualLength);   %pre-allocation

        for n=1:residualLength
```

```
            residual(n) = dot(lpcCov,fliplr(three_frames(n:n+usedP)));
        end

     %indices of residual overlaping frames(2,:):[frameLengthSamples−P]
        sourceFrame = residual(relativeIndexLeft+GCIs(ii):...
                relativeIndexRight+GCIs(ii));
        % Prosody normalise (x and y) the glottal estimate
        sourceFrame = ProsodyNormalise(sourceFrame,normdL);
        sourceFrames = [sourceFrames;sourceFrame]; % and store
    end
end % end of pitch periods loop
end % END OF FUNCTION
```

```
function [sourceFrame] = ProsodyNormalise(rawSourceFrame,normdL)

%resample(decimation or interpolation with required low pass filtering)
lengthOriginal = length(rawSourceFrame);
sourceFrame = resample(rawSourceFrame,normdL,lengthOriginal);

%also normalise the energy/amplitudes
scale = std(sourceFrame);
sourceFrame = sourceFrame/scale;

end
```