Lab 2 : LPC Analysis and Synthesis

September, 2006

1 Overview

This lab involves building an LPC (Linear Predictive Coding) system using Simulink. LPC is a signal processing tool used to represent the spectral envelope of speech in compressed form using the Linear Predictive model.

2 Theory

Speech signal consist of resonances created in the vocal tract and hisses created by the tongue, lips and throat. These resonances are called formants and the hisses are called buzz.

LPC analyzes the speech signal by estimating the formants, removing their effects from the speech signal, and estimating the intensity and frequency of the remaining buzz. The process of removing the formants is called inverse filtering, and the remaining signal after the subtraction of the filtered modeled signal is called the residue.

3 Implementation

This lab consists of two parts, analysis and synthesis. The analysis part is found in the transmitter section of a system. Using the original speech signal, reflection coefficients and the residual signal are extracted from it, and then transmitted over a channel. The synthesis part, which is found in the receiver section of the system, reconstructs the original signal using the reflection coefficients and the residual signal. In this experiment, the speech signal is divided into frames of size 20 ms (160 samples), with an overlap of 10 ms (80 samples). Each frame is then windowed using a hamming window, 11th order autocorrelation coefficients are found and then the reflection coefficients (11th order) are calculated from the autocorrelation coefficients, using the Levinson-Durbin algorithm. The original speech signal is passed through an analysis-filter which is an all-zero filter with coefficients as the reflection coefficients obtained earlier. The output of the filter is the residual signal. This residual signal is then passed through a synthesis-filter which is an all-pole filter, again with coefficients as the reflection
coefficients (the Synthesis filter is the inverse of the Analysis filter). The output of the synthesis filter is the original signal.

4 Software overview

In this lab, Simulink is used to model the system. This model is quite difficult to build from scratch. It is easier to work with the completed model.

5 Initial configuration

This section is the same as section 3 in Lab 2.

5.1 Check if CCS is properly installed

To verify that CCS is properly installed on the system, enter

```
ccsboardinfo
```

at the Matlab command line. Matlab should return information similar to the following listing:

<table>
<thead>
<tr>
<th>Board Num</th>
<th>Board Name</th>
<th>Proc Num</th>
<th>Processor Name</th>
<th>Processor Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C6713 DSK</td>
<td>0</td>
<td>CPU_1</td>
<td>TMS320C6x1x</td>
</tr>
</tbody>
</table>

To ensure Embedded Target for TI C6000 DSP is installed, enter

```
c6000lib
```

Matlab should display the C6000 block library containing the libraries: C6000 DSP Core Support, C62x DSP Library, C64x DSP Library, C6416 DSK Board Support, C6701 EVM Board Support, C6711 DSK Board Support, C6713 DSK Board Support, RTDX Instrumentation, TMDX326040 Daughtercard Support.

5.2 Configuration Parameters for C6000 Hardware

1. Launch Matlab

2. At the Matlab command line, type

```
simulink
```

to launch Simulink

3. Create a new model in Simulink.
4. To open the **Configuration Parameters**, select **Simulation**→**Configuration Parameters**.

5. In the **Select** tree, choose the **Real-Time Workshop** category.

6. For **Target Selection**, choose the file ti_c6000.tlc. Real-Time Workshop will automatically change the **Make command** and **Template makefile** selections.

7. Choose the **Optimization** category in the **Select** tree. For **Simulation and Code generation**, un-select **Block reduction optimization** and **Implement logic signals**.

8. Choose the **TI C6000 target selection**. Set **Code generation target type** to C6713DSK.

9. Choose the **TI C6000 compiler**. Set **Symbolic debugging** and uncheck **DSP/Bio Option**.

10. In the **Select** tree, choose the **Debug** category. Select **Verbose build** here.

11. In the **Select** tree, choose the **Solver** category. Ensure that **Solver** is set to **Fixed type / discrete**.

### 6 Building the LPC system

The block diagram that implements this system is shown in the figure. This experiment uses the .mat file mtlb.mat.

1. Load the mtlb.mat file with

   ```matlab
   load mtlb
   ```

2. For the **Signal from Workspace** block, use the following parameters:
   - Set sample time to 1/8000.
   - Set Samples per frames to 80.

3. For the **Pre-emphasis digital filter** block, change the following parameters:
   - Transfer function type: FIR (all zeros)
   - Filter structure: Direct form
   - Numerator coefficients: [1 -.95]

4. For the **Buffer** block, change the following parameters:
   - Output buffer size: 160
• Buffer overlap : 80

5. For the Autocorrelation block, change the Maximum non-negative lag to 12 and Scaling to biased.

6. For the Time varying analysis filter block, use the following parameters:
   • Transfer function type : FIR (all zeros)
   • Filter structure : Lattice MA
   • Coefficient source : Input port(s)
   • Coefficient update rate : One filter per frame

7. The Time varying synthesis filter block has the same parameters as the analysis filter block, except that the Transfer function type is IIR (all poles) and the Filter structure is Lattice AR.

8. The De-emphasis filter is an IIR filter with the same parameters as the Pre-emphasis filter.

9. Connect speakers to the Line Out connector of the DSK

10. The model is now complete and ready to run. Verify that the DSK board is connected properly. Use the Incremental Build command on the Simulink model toolbar (the icon with the 3 arrows) to begin compiling the model and transferring the code to the DSK.

11. If the model has downloaded successfully to the board, you should hear 'Matlab' through the speakers.
Figure 2: LPC Synthesis of Speech