Implementing digital IIR filters in the lattice form

Lab 6, SDP

Objective

The students should become familiar with *lattice*-type realization structure used for implementing IIR filters.

Theoretical notions

Exercises

1. Consider the causal IIR system with poles and zeros, with the system function:

$$H(z) = \frac{1 + 2z^{-1} + 3z^{-2} + 2z^{-3}}{1 + \frac{2}{5}z^{-1} + \frac{7}{20}z^{-2} + \frac{1}{2}z^{-3}}$$

   Find and draw the equivalent *lattice* structure for the IIR filter.

2. Consider the causal IIR system, with no zeros, with the following system function:

$$H(z) = \frac{1}{1 + \frac{2}{5}z^{-1} + \frac{7}{20}z^{-2} + \frac{1}{2}z^{-3}}$$

   Find and draw the equivalent *lattice* structure for the IIR filter.

3. In the Matlab environment, use the `fdatool` tool to design one of the following filters:

   a. A low-pass IIR filter of order 4, elliptic type, with cutoff frequency of 6kHz at a sampling frequency of 44.1kHz;

b. A high-pass IIR filter of order 4, elliptic type, with cutoff frequency of 2.5kHz at a sampling frequency of 44.1kHz;
c. A band-pass IIR filter of order 4, elliptic type, with passband between 0.5kHz and 5.5kHz at a sampling frequency of 44.1kHz.

4. In the Simulink environment, implement the above filters in *lattice* form. Apply at the input an audio signal and play the output signal, as well as the original, for comparison. How does the filtered signal sound like, compared to the original?
5. Create an Octave function to filter an input signal `x` with an IIR filter in lattice form, given the coefficients K and V:

   ```
   y = filter_latc_iir(K, V, x)
   ```

   Define variables `w1`, `w2`, ... to hold the values of the unit delays, and `w1_next`, ... to hold the future values.

   - Compute the current output value based on `w1`, ... and the input
   - Compute the next values `w1_next`, ... based on `w1`, ... and the input
   - Update `w1`, ... with the values in `w1_next`, ... and iterate

6. Use the function above to filter an audio file.

   a) Load the file using `audioread()`
   b) Filter the signal using `filter_latc_iir()`, with the previously designed filter

Notes:

- Set the following parameters for the SImulink model, to enable a discrete simulation with fixed (auto) step:
  - Type: *Fixed-step*
  - Solver: *discrete (no continuous states)*

- You will need the blocks *Unit Delay*, *Sum* and *Gain*
- At the input put a *From Multimedia File* block, and at the output put a *To Audio Device* block
- At the output, before the *To Audio Device* block, put a *Manual Switch* block in order to be able to switch easily between the original signal and the filtered one
- For the *From Multimedia File* block, select an audio file (de ex. Kalimba.mp3 from My Documents) and update the following settingsȘ
  - choose *Sample-based*
  - *Samples per audio channel* = 1
  - "DataTypes/Audio output data type" = *double*

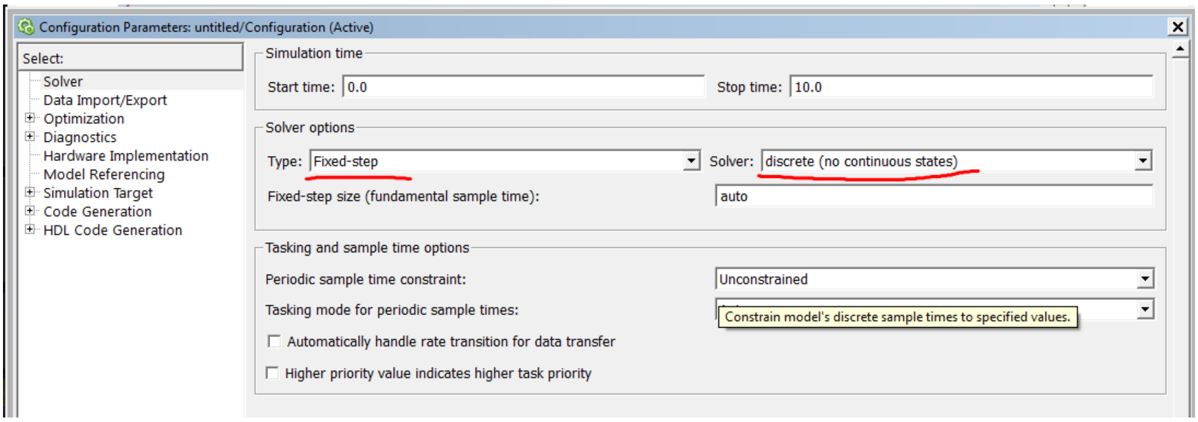
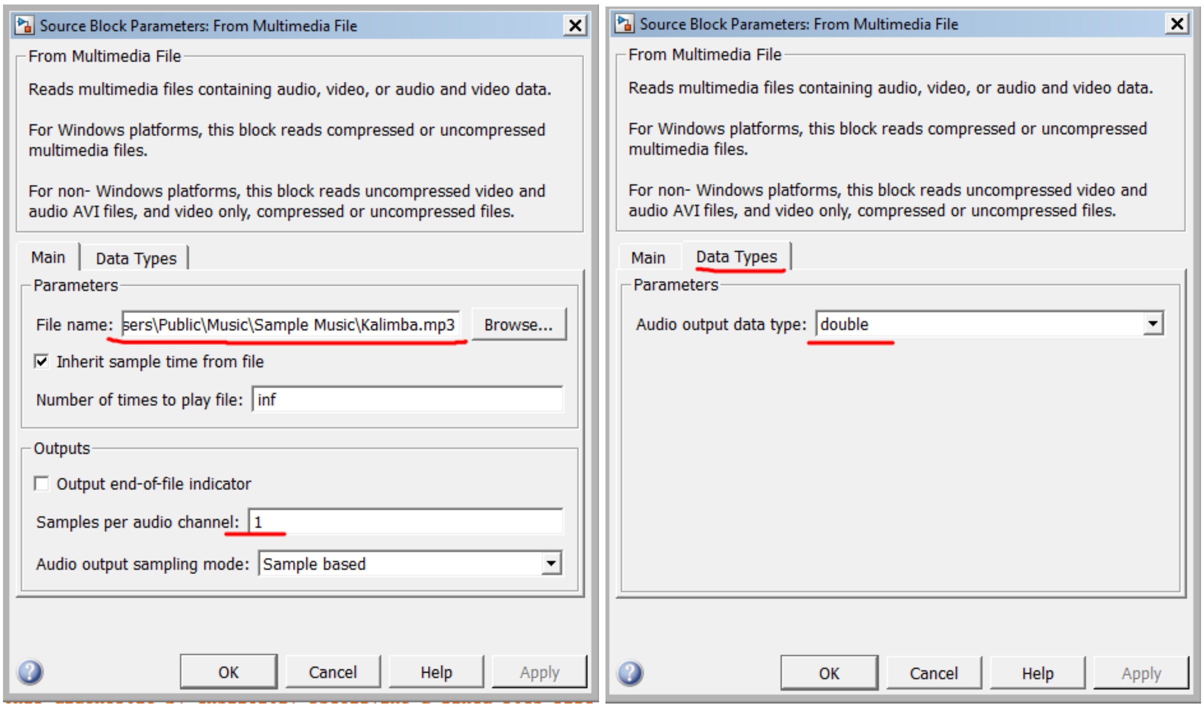


Figure 1: Model settings for discrete models



Final questions

1. TBD