

## 0.1 Low Pass Filter

The transfer function of the low pass filter is :

$$H(s) = \frac{1}{1 + sRC}$$

Discretizing using Tustin's approximation,  $s \leftarrow \frac{2}{T_s} \frac{1-z^{-1}}{1+z^{-1}}$

$$H(z) = \frac{T_s(1 + z^{-1})}{T_s(1 + z^{-1}) + 2(1 - z^{-1})RC}$$

If  $x(t)$  be the input to the filter and  $y(t)$  be the output produced by the filter, the above filter in time domain is :

$$y(t) = \frac{1}{T_s + 2RC} \left[ T_s \left( x(t) + x(t-1) \right) + (2RC - T_s)y(t-1) \right]$$

We can easily implement the above filter's form in code. Here  $T_s$  is the sampling period.

### 0.1.1 Implementation in MATLAB

---

```
% Transfer Function of RC circuit
% H = 1 / (1 + sRC)

function [out] = tust_lpf(in, time, fc)
% Apply Low Pass Filter using Tustin's Approximation

    out = zeros(size(in));
    out(1) = in(1);

    RC = 1 / (2*pi*fc);

    for i = 2:length(in)
        dt = time(i) - time(i-1);
        a = 1 / (dt + 2*RC);
        b = (2*RC - dt);
        out(i) = a * (dt * (in(i) + in(i-1)) ...
                    + b * out(i-1));
    end

end
```

---

## 0.2 Average Energy

We calculated the average energy in a given window. We used the rolling technique so that the chances of missing out any data is less. We used the *Parseval's Theorem* to calculate the average energy over the window.

According to *Parseval's Theorem*, if  $x[k]$  and  $X[f]$  are the pair of discrete time Fourier sequences, where  $x[k]$  is the discrete time sequence and  $X[f]$  is its corresponding DFT, the energy of the aperiodic sequence of length  $N$  can be expressed in terms of its  $N$ -point DFT as follows:

$$E_x = \frac{1}{N} \sum_{f=0}^{N-1} \left| X[f] \right|^2$$

### 0.2.1 Implementation in MATLAB

---

```

% Apply Parseval's Theorem to calculate the
% instantaneous energy(power) of the input signal

function [pars] = Energy(in, win_sz, roll_factor)
% Gives the instantaneous energy (power) of the
% signal according to the win_sz and the roll_factor

    roll_sz = floor(win_sz * roll_factor);

    n = length(in);
    pars = zeros(size(in));

    % Make 'n' a multiple of window size
    n = win_sz * floor(n / win_sz);

    for i = 1:roll_sz:(n - win_sz)
        x = in(i : i+win_sz);

        y = fft(x);    % Apply Fast Fourier Transform
        amp = abs(y);  % Get coefficients after
                        % transformation

        % Calculate Power using Parseval's Theorem
        % E = sum |X(f)|^2
        % P = (1 / N) * E
        pars(i) = amp' * amp / win_sz;

    end

end

```

---