Finite-Difference Equations with Constant Coefficients

PREFACE: The z-transform allows us to link the impulse responses that we have covered in this class with their respective finite-difference-equation (FDE) representations. These FDE's have constant coefficients, so their integrated response will match the original impulse response, and application of the FDE to an input signal will match the result of convolution. In lecture #6 I will show that FDE's with dynamic coefficients generate an even broader class of regularizing filters, as long as they remain stable.

This homework focuses on the connection between our family of impulse responses and the FDE representation.

PREPARATION: At a high level, you will need to write the FDE that corresponds to each impulse response $h[n]$, implement the recursion, match its impulse response to $h[n]$, and then apply the recursion to pricing data.

PROBLEMS:

1. **FDE Representation:** Derive analytically the finite-difference equation that corresponds to each impulse response $h[n]$. In your solutions give the z-transform $H(z)$ for each $h[n]$, and recognizing that each transform is a ratio of polynomials in z, $H(z) = P(z)/Q(z)$, give $P(z)Y(z) = Q(z)X(z)$. Finally, take the inverse z-transform and give the FDE that expresses $y[n]$ in terms of $x[n]$.

   (a) Delayed Impulse: $h[n] = \delta[n - N]$.

   (b) Unit Step: $h[n] = u[n]$.

   (c) Box: $h_{\text{box}}[n] = N_{\text{box}}^{-1}\,(u[n] - \delta[n - N_{\text{box}}] * u[n])$.

   (d) Ema: $h_{\text{ema}}[n] = (1-p)p^n u[n]$.

   (e) Ema-Poly1: $h_{\text{poly}}[n] = (1-p)^2(n+1)p^n u[n]$.

   (f) Lifted Macd-Poly:

$$h_{\text{lift}}[n] = (2N_{\text{eff}}/3)^{-1}\; u[n] * (h_{\text{ema}}(N_{\text{eff}}/3)[n] - h_{\text{poly}}(N_{\text{eff}})[n])$$

   (g) Macd: $h_{\text{macd}}[n] = h_{\text{ema}}(N_{\text{eff}\,+})[n] - h_{\text{ema}}(N_{\text{eff}\,-})[n]$.

2. **FDE Implementation:** For each impulse response write a routine in Matlab or R that implements the recursion. The function signature should resemble

```
y = apply_ema_filter(x, N_eff)
```

That is, each apply_*() function should consume x, a vector input, and all the necessary parameters to define the filter characteristics. For an ema the necessary parameter is N_{eff}. Within the function the filter parameter(s) (such as N_{eff}) are to be converted FDE parameters (such as p), and finally to recursion coefficients.

In order to implement these recursions correctly you must consider the initial conditions. In general a recursion equation is written as

$$\begin{aligned} y[n] = \phi_1 y[n-1] + \phi_2 y[n-2] + \cdots \\ + \theta_0 x[n] + \theta_1 x[n-1] + \cdots \end{aligned}$$

There are ℓ_y lags of y and ℓ_x lags of x[1]. To start at $n = 0$ you need initial conditions $y[-1], y[-2], \cdots$. For this homework, force all lags of $y[n], n < 0$ to zero; that is, the initial conditions for $y[n]$ are zero. A simple way to do this is to pad $x[n]$. Given ℓ_y lags of y, define $\tilde{x}$ as

```
x_tilde  = [zeros(lags_y,1); x(:)];
```

where I have used matlab notation. Then initialize $\tilde{y}$ as

```
y_tilde  = zeros(size(x_tilde));
```

Run the recursion with a `for` loop but start the index at $\ell_y + 1$. Once the recursion is complete, remove the padded section and return y as in

```
y = y_tilde(n_lag_y+1;end)
```

You need to also address the first value of $x[n]$. Recall from homework #2 that for convolution with unity-gain filters you need to subtract $x[0]$ from the series, convolve, then add $x[0]$ back. The same holds true here. So, for unity gain filters, when padding $\tilde{x}$ also subtract the first input value:

```
x_tilde  = [zeros(lags_y,1); x(:) - x(1)];
```

and add $x(1)$[2] back at the end. For the unit-step, the gain is infinite so don't subtract or add $x(1)$. For the macd, the gain is zero so don't add $x(1)$ at the end, but subtract it at the beginning.

3. **Verify the FDE Impulse Response:** Whenever you implement an FDE with constant coefficients that is intended to represent an impulse response $h[n]$ you should always verify that the code you have written does indeed generate $h[n]$.

   To do this you need to input an impulse and record the output. The location of the impulse at the input requires some attention. The short answer is that you should use

$$x[n] \equiv \delta[n - \ell_y]$$

[1] Note that $y[n]$ is not a lag of y, nor is $x[n]$ a lag of x.
[2] using matlab notation, and matlab is one based.

so that $x[\ell_y] = 1$ and is zero otherwise. The reason to use this delayed impulse is because, for filters other than delay and unit-step, $x[0]$ is subtracted from $x[n]$, so $x[n] = \delta[n]$ would actually input a series of -1's into the recursion. Also, you will need to crop the output by removing the first ℓ_y entries.

For this problem, input $x[n] \equiv \delta[n-\ell_y]$ into each FDE function and record the output $h_{\text{fde}}[n]$. Use the filter parameters that follow. Also, separately calculate $h[n]$ using the code you wrote for homeworks 2 and 3 using the same parameters. For each filter overlay $h_{\text{fde}}[n]$ and $h[n]$, one plot for each filter type. The impulses responses should be identical.

(a) Delay: $N_{\text{delay}} = 32$

(b) Box: $N_{\text{box}} = 32$

(c) Ema: $N_{\text{eff}} = 32$

(d) Ema-Poly1: $N_{\text{eff}} = 32$

(e) Lifted Macd-Poly: $N_{\text{eff}} = 32$ for the FDE, $N_{\text{eff}} = (24/23) \times 32$ for the make_h_lifted_macd_poly() function from homework #3.

(f) Macd: $N_{\text{eff}\,+} = 16$, $N_{\text{eff}\,-} = 32$

and in general $N_{\text{window}} = 8 \times 32$.

4. **Apply FDE's to a Price Series, Compare with Convolution:** For each FDE filter, input the price series from JPM that you generated in homework #2 and record the result as (e.g.) $y_{\text{fde}}[n]$. Also, convolve the price series with each impulse response $h[n]$ and record the result as $y_h[n]$. For each filter overlay $y_{\text{fde}}[n]$ and $y_h[n]$, one plot for each filter type. Use the previous filter parameters. The output series should be identical.

Homework #5 Solutions

PROBLEMS:

FDE Equations

1a Delayed Impulse:

$$h[n] = \delta[n - N]$$
$$H(z) = z^{-N}$$
$$Y(z) = z^{-N} X(z)$$
$$y[n] = x[n - N]$$

1b Unit Step:

$$h[n] = u[n]$$
$$H(z) = \frac{1}{1 - z^{-1}}$$
$$\left(1 - z^{-1}\right) Y(z) = X(z)$$
$$y[n] = y[n - 1] + x[n]$$

1c Box:

$$h[n] = N_{\text{box}}^{-1} \left(u[n] - \delta[n - N_{\text{box}}] * u[n]\right)$$
$$H(z) = N_{\text{box}}^{-1} \frac{1 - z^{-N_{\text{box}}}}{1 - z^{-1}}$$
$$\left(1 - z^{-1}\right) Y(z) = N_{\text{box}}^{-1} \left(1 - z^{-N_{\text{box}}}\right) X(z)$$
$$y[n] = y[n - 1] + N_{\text{box}}^{-1}\, x[n] - N_{\text{box}}^{-1}\, x[n - N]$$

1d Ema:

$$h[n] = (1 - p)p^n u[n]$$
$$H(z) = \frac{1 - p}{1 - pz^{-1}}$$
$$\left(1 - pz^{-1}\right) Y(z) = (1 - p)\, X(z)$$
$$y[n] = py[n - 1] + (1 - p)\, x[n]$$

1e Ema-Poly1:

$$h[n] = (1-p)^2(n+1)p^n u[n]$$
$$H(z) = \frac{(1-p)^2}{(1-pz^{-1})^2}$$
$$\left(1 - 2pz^{-1} + p^2z^{-2}\right) Y(z) = (1-p)^2 X(z)$$
$$y[n] = 2p\, y[n-1] - p^2 y[n-2] + (1-p)^2 x[n]$$

1f Lifted Macd-Poly:

$$h[n] = (2N_{\text{eff}}/3)^{-1}\, u[n] * (h_{\text{ema}}(N_{\text{eff}}/3)[n] - h_{\text{poly}}(N_{\text{eff}})[n])$$
$$H(z) = \frac{(2N_{\text{eff}}/3)^{-1}}{1-z^{-1}} \left(\frac{1-p_{\text{ema}}}{1-p_{\text{ema}}z^{-1}} - \frac{(1-p_{\text{poly}})^2}{(1-p_{\text{poly}}z^{-1})^2} \right)$$

$$\left(1 - (2p_{\text{poly}} + p_{\text{ema}})z^{-1} + p_{\text{poly}}(p_{\text{poly}} + 2p_{\text{ema}})z^{-2} - p_{\text{poly}}^2 p_{\text{ema}} z^{-3}\right) Y(z) =$$
$$\left((2p_{\text{poly}} - p_{\text{poly}}^2 - p_{\text{ema}}) - p_{\text{poly}}^2(1-p_{\text{ema}})z^{-1}\right) X(z)$$

$$y[n] = (2p_{\text{poly}} + p_{\text{ema}})y[n-1] - p_{\text{poly}}(p_{\text{poly}} + 2p_{\text{ema}})y[n-2] + p_{\text{poly}}^2 p_{\text{ema}} y[n-3]$$
$$+ (2p_{\text{poly}} - p_{\text{poly}}^2 - p_{\text{ema}})x[n] - p_{\text{poly}}^2(1-p_{\text{ema}})x[n-1]$$

1g Macd:

$$h[n] = h_{\text{ema}}(N_{\text{eff}\,+})[n] - h_{\text{ema}}(N_{\text{eff}\,-})[n]$$
$$H(z) = \frac{1-p_+}{1-p_+z^{-1}} - \frac{1-p_-}{1-p_+z^{-1}}$$
$$\left(1 - (p_- + p_+)z^{-1} + p_-p_+\, z^{-1}\right) Y(z) = \left((p_+ - p_-) - (p_+ - p_-)z^{-1}\right) X(z)$$
$$y[n] = (p_- + p_+)y[n-1] - p_-p_+\, y[n-2] + (p_- - p_+)x[n] - (p_- - p_+)x[n-1]$$

2 See apply_∗ _filter .m scripts.

3 See study_fde_impulse_responses.m

4 See study_fde_application .m