ECE551, Fall 2016: Homework Problem Set #7
Due Nov. 8 2018 by 12:30pm

1. **Truncation as Filter Approximation**
We define the *truncation* operator $T_I : \mathbb{C}^{\mathbb{Z}} \to \mathbb{C}^{\mathbb{Z}}$ associated an index subset $I \subset \mathbb{Z}$ as $(T_I h)[t] := w[t]h[t]$, where w is the window (indicator) function of I

   (a) Let $h_d \in \ell_2(\mathbb{Z})$ be some (desired) filter that we want to approximate on $\ell_2(I)$. Show that the truncation $\hat{h} := T_I h_d$ yields the least-squares approximation of h^d in $\ell_2(I)$. You may use the orthogonality principle.
   (b) Show that T_I is an orthogonal projection on $\ell_2(I)$.
   (c) Find the best FIR approximation of the lowpass $h_d[n] = (\operatorname{sinc} \frac{\pi}{3} n)$ on $I = \{0, \ldots, 4\}$.
   (d) Repeat the last part for an FIR of length $L = 5$ (choose the window position).

2. **Lagrange Interpolation**
Let $D := \{(t_k, x_k)\}_{k=0}^{N-1}$ be a set of points on the plane such that $t_k \neq t_j$, and let $p_D(t)$ be a polynomial of degree $N-1$ that interpolates on D:

$$p_D(t_k) = x_k \qquad \text{for all } k = 0, \ldots, N-1 \tag{1}$$

   (a) Suppose that we add a new point (t_N, x_N) to D and let $\tilde{D} = D \cup \{(t_N, x_N)\}$ (you may assume that $t_N \neq t_k$). Find a constant c for which

$$p_{\tilde{D}}(t) = p_D(t) + c(t - t_0)(t - t_1) \cdots (t - t_{N-1}) \tag{2}$$

   (b) Use the result above to show that

$$p_D(t) = \sum_{k=1}^{N-1} x_k \prod_{i \neq k} \frac{t - t_i}{t_k - t_i} \tag{3}$$

3. **Polynomial Spaces with Orthogonality**
Consider the space $\mathbb{C}[t]$ of all polynomials in t, equipped with some inner product $\langle \cdot, \cdot \rangle$. Let $\{v_0, v_1, ...\} \subset \mathbb{C}[t]$ be a sequence of nonzero polynomials satisfying

$$\langle v_i, v_k \rangle = \delta[k - j] \qquad \text{for all } k, j \geq 0 \qquad \text{(orthogonality)}$$
$$\deg(v_k) \leq k \qquad \text{for all } k \geq 0 \qquad \text{(degree)}$$

We define for all $n \geq 0$ the following two subspaces of $\mathbb{C}[t]$:

$$V_n := \operatorname{span}\{v_0, \ldots, v_n\} \tag{4}$$
$$W_n := \operatorname{span}\{1, t, t^2, \ldots, t^n\} \tag{5}$$

   (a) Show that $V_n = W_n$. **Hint:** show that $V_n \subset W_n$ and find their dimensions.
   (b) Show that any $p \in \mathbb{C}[t]$ of degree m must be orthogonal to all v_k of $k > m$.
   (c) Show that V_n is shift-invariant in t: for every $v \in V_n$ we have $v(\cdot - t_0) \in V_n$.

4. **Polynomial Spaces vs. Spline Spaces**
For a real function ϕ (referred to as *a template*), we define the space of signals *generated by shifts* of ϕ as

$$U := \mathrm{span}_{n\in\mathbb{Z}}\{\phi(\cdot - nT)\} \tag{6}$$

(a) Suppose that $T = 1$ and that ϕ is a triangular window: $\phi(t) = \begin{cases} 1-|t| & t \in [-1,1] \\ 0 & \text{else} \end{cases}$.
Sketch the graphs (use Python if you wish) of the signals $s_0, s_1 \in U$ given by

$$s_0(t) = \sum_{n=-N}^{N} \phi(t-n) \qquad\qquad s_1(t) = \sum_{n=-N}^{N} n\phi(t-n)$$

(b) Show that the polynomials $p_0(t) = 1$ and $p_1(t) = t$ are contained (as a limit) in U. **Hint:** first prove that $a[n] = \sum_{k\in\mathbb{Z}} a[k]\beta(k-n)$.

(c) From the last part we see that u_0, u_1 belong to U as well as to V_1 of (4), both of which are shift invariant spaces. Can we claim that $U = V_1$? explain.

5. **Interpolation with Shifted Symmetric Functions**
Let U be a space defined by (6), where the template $\phi(t) = 0$ outside the interval $[-NT, NT]$ where $N \geq 1$ is some integer and $T > 0$ is a real number.

Given a sequence of values $\{x[n]\}$, we want to find some $s \in U$, having the form

$$s(t) = \sum_{k\in\mathbb{Z}} x[k]\phi(t-kT), \tag{interpolating function}$$

that interpolates between the values of x on $t = nT$:

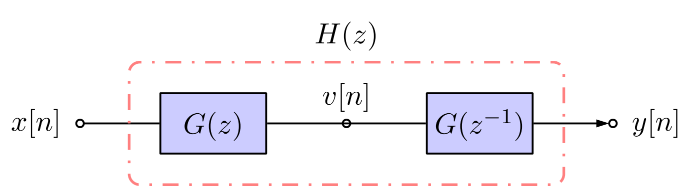
$$s(nT) = x[n] \quad \text{for all } n \in \mathbb{Z} \tag{interpolation condition}$$

Note: we have discussed this problem in the context of B-spline templates, but this analysis is valid for a much larger family of templates.

(a) Write the interpolation condition as a convolution sum, and derive a convolution filter to compute the coefficient sequence $\{c[k]\}$ in the Z-domain. That is, find $H(z)$ such that $C(z) = H(z)X(z)$. Determine what condition on ϕ is necessary to enable that.

(b) Assuming symmetric template, i.e $\phi(t) = \phi(-t)$, show that the roots/poles of $H(z)$ you derived are reciprocal: if $z = \lambda$ is a pole then so is $z = \lambda^{-1}$.

(c) Assuming that $H(z)$ has no poles with $|z| = 1$, show that it can be factored to $H(z) = G(z)G(z^{-1})$ where $G(z)$ is a causal-stable filter (with poles having $|z| < 1$).



(d) Sketch a diagram (of sums, shifts, and gains) of $H(z)$ as a cascade of causal $G(z)$ and anti-causal part $G(z^{-1})$.

6. **Python: Interpolation Games**

We revisit problem 5 with $T = 1$ and the templates:

$$\phi_0(t) = \beta^{(0)}(t) \qquad \phi_1(t) = \beta^{(1)}(t) \qquad \phi_2(t) = \beta^{(2)}(t)$$

$$\phi_3(t) = \begin{cases} \frac{1}{2}(1+\cos(\pi t)) & |t| \leq 1 \\ 0 & \text{else} \end{cases} \qquad \phi_4(t) = \begin{cases} \cos(\frac{\pi}{2}t) & |t| \leq 1 \\ 0 & \text{else} \end{cases}$$

here $\beta^{(K)}(t)$ is a uniform B-spline of order K as defined in part 6.3.2 of the textbook.

(a) Complete the following code, implementing a stable IIR cascade $x \to v \to y$ where $v[n+1] = \mu v[n] + \gamma x[n]$ and $y[n] = \mu y[n+1] + \gamma v[n]$:

```
def fwd_bwd_filter(gamma,mu,x):
    v,y  = np.zeros(len(x)), np.zeros(len(x))
    v[0] = ? # Based on your chosen boundary conditions
    for t in range(1,len(x)):
        v[t] = ?
    y[len(y)-1] = ? # Based on your boundary conditions
    for t in range(len(x)-1,0,-1):
        y[t-1] = ?
    return y
```

(b) Complete the following code. The goal is to find an interpolating $s_k(t)$ for each $\{\phi_k\}$, (i.e determine the coefficients $\{c[n]\}$) such that $s(n) =$ the n'th digit of your UIN, $1 \leq n \leq 9$. Comment on continuity/differentiability on the plots.

❀ Tip

In this problem we use `lambda` syntax, which is an alternative way to define simple functions in Python. What would traditionally be

```
def phi(t):
    return 1-abs(t)
```

becomes in lambda-form:

```
phi = lambda t: 1-abs(t)
print(phi(0), phi(-1)) # Will print 1,0
```

That way, we can have an array of functions (see code below).

```
import numpy as np
import matplotlib.pyplot as plt
from numpy import roots,zeros, sqrt, linspace, cos, pi, arange
phi = [ lambda t: ?, # phi_0
        lambda t: (1-abs(t))*(abs(t)<1), # phi_1
        lambda t: ?, # phi_2 return b-spline with K=2 at the values of t
        lambda t: cos(pi*t)*(abs(t)<0.5), # phi_3
        lambda t: ?] # phi_4
```

```
filters = [ lambda x: x,
               lambda x: ?,
               lambda x: fwd_bwd_filter(?,?,x),
               lambda x: ?,
               lambda x: ?,]

x = [your UIN]
N = 500
t = linspace(0,10,N)

for k in range(len(phi)):
    c = filters[k](x) # Compute the coefficients ...
    s =  # Compute the interpolating function here....

    plt.subplot(len(phi),1,k+1)
    plt.plot(t,s)
    plt.plot(arange(9),x,'rx')
```

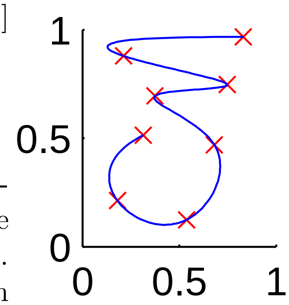
(c) Splines are often used in computer graphics to create smooth curves, passing through control points the artists feed (using their mouse). The following code graphically collects $N = 5$ mouse input points on the square $[0,1] \times [0,1]$. The returned array `points` is of size $N \times 2$, whose rows contain coordinates.

```
import numpy as np
import matplotlib.pyplot as plt
N = 5 # Number of points
plt.figure() # Open a figure()
plt.axis([0,1,0,1])  # ... and an axis
points = np.array(plt.ginput(N)) # Pick N points using mouse input
plt.plot(points.T[0],points.T[1],'rx') # Plot them
```

We look for a parametric curve $\rho(t) = \begin{bmatrix}\rho_x(t) & \rho_y(t)\end{bmatrix}$ on $t \in [1, N]$ that interpolates between the points, that is

$$\rho(n) = \begin{bmatrix}\texttt{points}[n,0], & \texttt{points}[n,1]\end{bmatrix}, \quad \text{for } n = 1, \ldots, N$$

Use your code from the previous problem to compute the interpolating coefficients $\{c_x[k], c_y[k]\}$ for all templates $\{\phi_k\}$. Plot the resulting curves on top of the points. Comment on your results. **Tip:** don't forget the `plot` command linearly interpolates between points (that can be exploited for ϕ_1).



Example with $N = 8$.