MIS Homework 4

Xinhao Du

February 2023

Problem A5.5

Solution:

a

const_basis = fourier_basis_const(n)
# Check the norm of the constant basis vector
norm(const_basis)
# Check the inner product between the constant basis vector
dot(const_basis, const_basis)
sin_basis = fourier_basis_sin(1, n)
# Check the norm of the sine basis vector
norm(sin_basis)
# Check the inner product of the constant basis vector with the sine basis vector
dot(const_basis, sin_basis)
# Check the inner product between the sine basis vector
dot(sin_basis, sin_basis)
# Check the norm of the cosine basis vector
cos_basis = fourier_basis_cos(1, n)
norm(cos_basis)
# Check the inner product between the cosine basis vector
dot(cos_basis, cos_basis)
# Check the inner product of the sine basis vector with the cosine basis vector
dot(sin_basis, cos_basis)

b

Using Plots
n = 50
$d = fourier\_basis\_const(n)$
$s_1 = fourier\_basis\_sin(1, n)$
$c_1 = fourier\_basis\_cos(1, n)$

$s_2 = fourier\_basis\_sin(2, n)$
$c_2 = fourier\_basis\_cos(2, n)$
$a = fourier\_basis\_alt(n)$
$plot(1 : n, d, label = "d")$
$plot!(1 : n, s1, label = "s_1")$
$plot!(1 : n, c1, label = "c_1")$
$plot!(1 : n, s2, label = "s_2")$
$plot!(1 : n, c2, label = "c_2")$
$plot!(1 : n, a, label = "a")$

c

n=6
$coefficient_d = dot(d, x))$
$coefficient_{s1} = dot(s_1, x)$
$coefficient_{c1} = dot(c_1, x)$
$coefficient_{s2} = dot(s_2, x)$
$coefficient_{c2} = dot(c_2, x)$
$coefficient_a = dot(a, x)$
$x' = coefficient_d * d + coefficient_{s1} * s_1 + coefficient_{c1} * c_1 + coefficient_{c2} * c_2 + coefficient_{s2} * s_2 + coefficient_a * a$
Then, we can have that x'=[1,2,3,4,5,6].

Problem 6.18

Solution:

6.18

An mxn Vandermonde matrix are linearly dependent when an n-vector C exists,

$Vc = \sum_{i=1}^{n} C_i V_i = 0$, all entries of C are not 0. The entries of Vc are $p(t) = C_1 + C_2 t + \cdots + C_n t^{n-1}$ when $t_1 \cdots t_m$.

Using the fact from hint, when $Vc=0$, than $C=0$, so the columns of V are linearly independent.

Problem 6.22

Solution:

a

Adding A and B costs mn flops, adding x and y costs n flops, and the matrix-vector multiplying costs 2mn flops, so the total is 3mn+n flops and the approximate flop is 3mn flops.

b

Each matrix-vector multiplication costs 2mn flops, so the total is 8mn flops. Three vector additions costs 3m flops, so the total is 8mn+3m flops and the approximate flop is 8mn flops.

c

The first method requires fewer flops.

Problem A6.2

Solution:

mis4.jl > ...

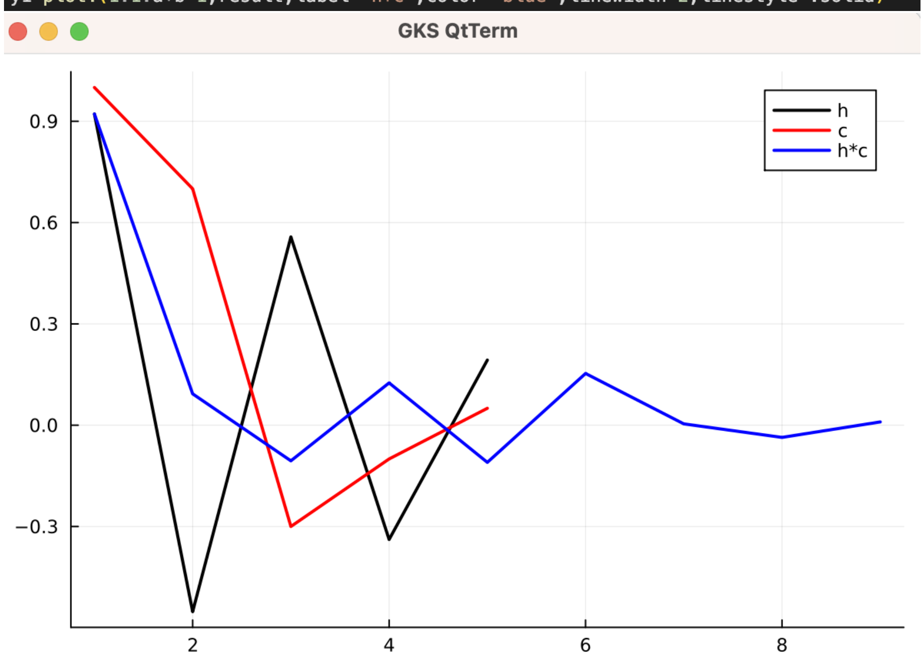
```julia
function Vandermonde(n,t::AbstractArray)
    m=length(t)
    tmp=map(tuple,t[i]^(j-1) for i=1:m,j=1:n)
    return tmp
end
# Vandermonde(4,rand(1:10,6))
println("Vandermonde :",Vandermonde(4,rand(1:10,6)))
```

```
6×4 Matrix{Tuple{Int64}}:
 (1,)  (1,)  (1,)   (1,)
 (1,)  (3,)  (9,)   (27,)
 (1,)  (2,)  (4,)   (8,)
 (1,)  (4,)  (16,)  (64,)
 (1,)  (4,)  (16,)  (64,)
 (1,)  (2,)  (4,)   (8,)
```

Problem A7.1

Solution:

```
include("channel_equalization_data.jl")
using Plots
import Plots:plot
a=length(h)
b=length(c)
result=zeros(a+b-1)
for i in 1:(a+b-1)
    for j in 1:a
        if i+1-j>=1 && i+1-j<=b
            result[i]+=h[j]*c[i+1-j]
        end
    end
end
display(result)
y1=plot(1:1:a,h,label="h",color="black",linewidth=2,linestyle=:solid)
y1=plot!(1:1:b,c,label="c",color="red",linewidth=2,linestyle=:solid)
y1=plot!(1:1:a+b-1,result,label="h*c",color="blue",linewidth=2,linestyle=:solid)
```



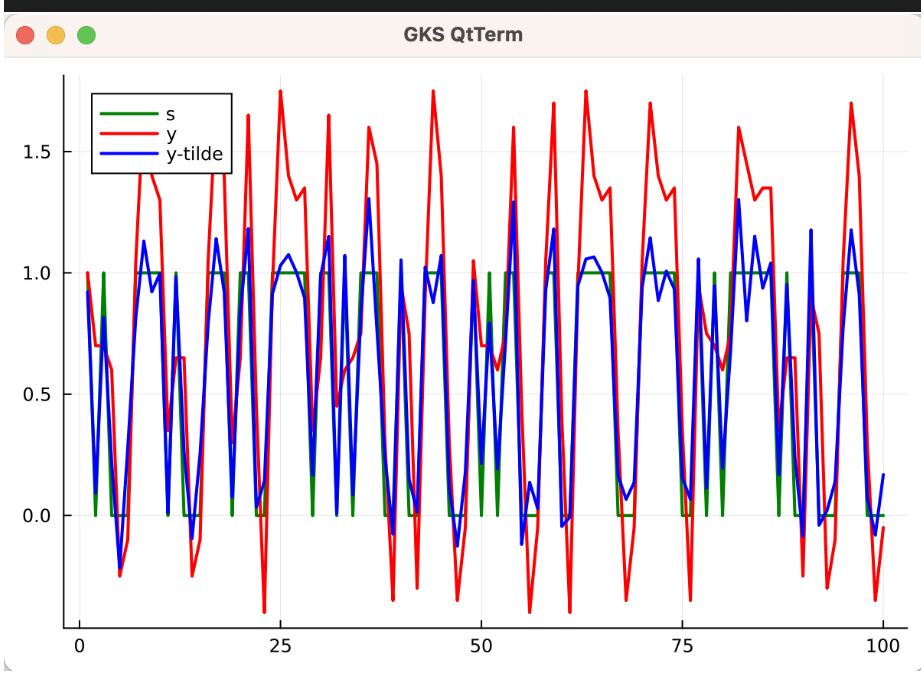
Chanel c is the system impulse response, and the plot of c shows that c waves are random. However, adding h, which had been bouncing up and down about line 0, and results in a longer and average serious impulse response, h*c is more equalized and stabled.

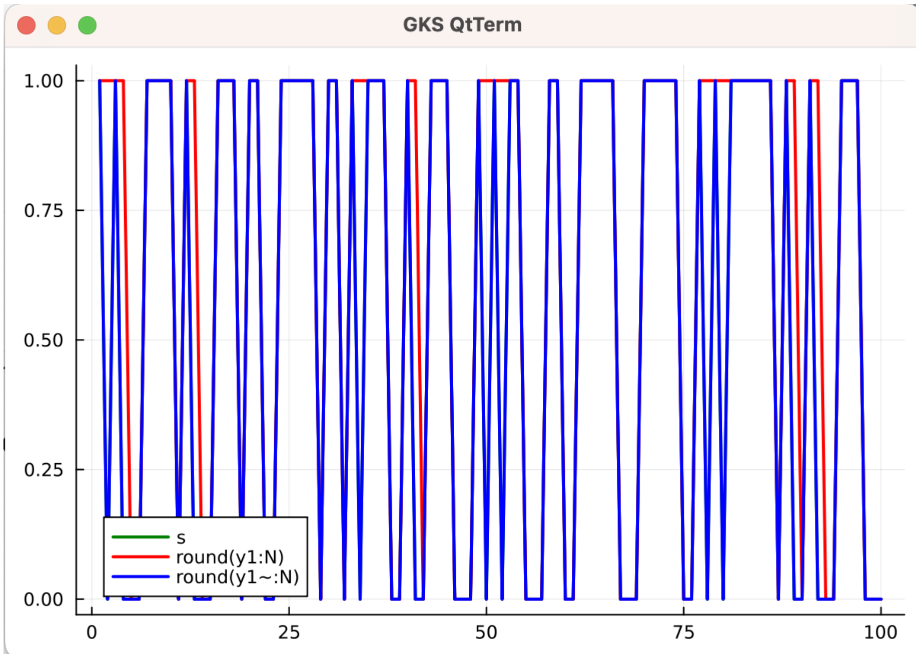
Then we plot s,y, and y-tilde

```
n=length(c)
m=length(result)
l=length(s)
y=zeros(l+n-1)
yt=zeros(m+l-1)
for i in 1:l+n-1
    for j in 1:n
        if i+1-j>=1 && i+1-j<=l
            y[i]+=c[j]*s[i+1-j]
        end
    end
end
for i in 1:l+m-1
    for j in 1:m
        if i+1-j>=1 && i+1-j<=l
            yt[i]+=result[j]*s[i+1-j]
        end
    end
end

y2=plot(1:1:100,s[1:100],label="s",color="green",linewidth=2,linestyle=:solid)
y2=plot!(1:1:100,y[1:100],label="y",color="red",linewidth=2,linestyle=:solid)
y2=plot!(1:1:100,yt[1:100],label="y-tilde",color="blue",linewidth=2,linestyle=:solid)

y=1*(y.>0.5)
yt=1*(yt.>.5)
y3=plot(1:1:100,s[1:100],label="s",color="green",linewidth=2,linestyle=:solid)
y3=plot!(1:1:100,y[1:100],label="round(y1:N)",color="red",linewidth=2,linestyle=:solid)
y3=plot!(1:1:100,yt[1:100],label="round(y1~:N)",color="blue",linewidth=2,linestyle=:solid)
```





Plots $S = round(y1 : N)$ and $S^{eq} = round(y1\ \ : N)$

It is clear that $s = round(y1 : N)$ is worse estimate of $s^{eq} = round(y1\ \ : N)$

```julia
BER1=0
BER2=0
for i in 1:length(s)
    if y[i]!=s[i]
        BER1+=1
    end
end
for i in 1:length(s)
    if yt[i]!=s[i]
        BER2+=1
    end
end
println("BER for s^: ",BER1/length(y))
println("BER for s^eq: ",BER2/length(y))
```

```
julia> println("BER for s^: ",BER1/length(y))
BER for s^: 0.11553784860557768

[julia> println("BER for s^eq: ",BER2/length(y))
BER for s^eq: 0.0
```

As we can see from the figure, the BER for s is 0.11553784860557768, and for s^{eq} is 0.

Problem A7.3

Solution:

a

```
using WAV
x,f=wavread("audio_filtering_original.wav");
x=vec(x)
wavplay(x,f)
```

```
h_smooth = 1 / 44 * ones(44);
output = conv(h_smooth, x);
wavplay(output, f);
```

The music is more peace and softer after convolving by h^{smooth}.

b

Since k is the number of samples in 0.25s, k=0.25*(441000/10)=11025, which means h-echo=(1,0,...0,0.5).

```
h_echo=zeros(11025)
h_echo[1]=1
h_echo[11025]=0.5
output1=conv(h_echo,x)
wavplay(output1,f)
h_echo2=conv(h_echo,h_echo)
output2=conv(h_echo2,x)
wavplay(output2,f)
```

We could hear two music, the first one is just like the original one, and the second one is weaker than the original music.

Problem 7.8

Solution:

$Ax + s = 0$

$1^T(Ax+s) = 0$

$(1^T A)x + (1^T s) = 0$

A is the incidence matrix, and $1^T A$ is the row vector of columns sums of A, so $1^T A = 0$.

Thus we can have $1^T s = 0$.