<!-- -->

Table of contents

1 Digital Filter Design Flash Demo..2
2 FrAid scripts used in the demo...2
2.1 Properties of smoothing filters...2
2.2 Compare different window functions...2
2.3 Compare filters with different lengths...3
2.4 Filter inverse, fiter reverse..3
2.5 Band-pass/Band-reject out of a High-Pass and a Low-Pass..3
2.6 Interactive test..4

1. Digital Filter Design Flash Demo

2. FrAid scripts used in the demo

2.1. Properties of smoothing filters

```
samplingF = 1024; stepT = 1/samplingF; seqL = 2; filterL = 5; start = 100;
controlVar(start,100, filterL,10); contF(x) = if x < start * stepT then 0 else if x < ( start
+ filterL ) * stepT then 1/integer(filterL) else 0; plot(contF, (start - 20) * stepT, 2, (start
+ filterL + 20) * stepT, -2 ); f(x)=sampleL(contF,0,seqL,seqL*samplingF); plot(f);
ff(x)=fft1(f); ffn(x)=ff(x)/maxS(ff); //normalize, or multiply by the samplingF?
ffp(x)=abs(ffn(x)); plot(ffp); //autoScale, should I divide by maxS(ffp)?
db(x)=8.685890*log(ffp(x)); plot(db); iff(x)=ifft1(ff); plot(iff); //new stuff
flr(x)=firResp(fl); //what is fl? plot(flr,-.1,1.1,Pi+.1,-.5);
```

2.2. Compare different window functions

```
samplingF = 2048; //Hz samplingTime = 1; //seconds filterLength = 41; //points
shape(x) = if x < ( samplingF / 2 ) / 3 then 1 else 0; //say limit to a third of the interval
fr(x)=sampleN( shape, 0, 1, samplingF/2+1 ); //freq. response //plot(fr); fk(x)=ifft1(fr);
//filter kernel //plot(fk); fktr(x)=truncateS(shrotS(fk,filterLength/2),0,filterLength-1);
fkn(x)=fktr(x)/sumS(fktr); //normalized kernel //plot(fkn); hammingW(x) = 0.54 - 0.46 *
cos( 2 * Pi * x / filterLength ); //Hamming blackmanW(x) = 0.42 - 0.5 * cos( 2 * Pi * x /
filterLength ) + 0.08 * cos( 4 * Pi * x / filterLength ); //Blackman
wfkh(x)=hammingW(x/stepS(fkn))*fkn(x); //the Hamming windowed filter
wfkb(x)=blackmanW(x/stepS(fkn))*fkn(x); //the Blackman windowed filter //check the
resulting filters frequency response efr(x)=firResp(fkn); //estimated freq. response
non-windowed efrh(x)=firResp(wfkh); //estimated freq. response Hamming
efrb(x)=firResp(wfkb); //estimated freq. response Blackman
plot(efr,efrh,efrb,-.1,1.1,Pi+.1,-.5); db(x)=8.685890*log(x); efrdb(x)=db(efr(x)); //same
as above but in db efrdbH(x)=db(efrh(x)); efrdbB(x)=db(efrb(x));
plot(efrdb,efrdbH,efrdbB,-.1,1,Pi+.1,-70); //alternatively do the same through the
fourier transform of the filter kernel
fknp(x)=padS(fkn,2^nextpow2(samplingF*samplingTime)); //pad to the proper length
fkn1f(x)=fft1(fknp); //take fft fkn1a(x)=abs(fkn1f(x))*samplingF/2; //take abs and
normalize fkn1db(x)=db(fkn1a(x)); //calculate in db
wfkp1(x)=padS(wfkh,2^nextpow2(samplingF*samplingTime)); //same for the
Hamming windowed kernel... wfkh1f(x)=fft1(wfkp1);
wfkh1a(x)=abs(wfkh1f(x))*samplingF/2; wfkh1db(x)=db(wfkh1a(x));
```

wfkbp(x)=padS(wfkb,2^nextpow2(samplingF*samplingTime)); //same for the Blackman windowed kernel... wfkb1f(x)=fft1(wfkbp); wfkb1a(x)=abs(wfkb1f(x))*samplingF/2; wfkb1db(x)=db(wfkb1a(x)); //plot(fkn1,wfkh1,wfkb1); plot(fkn1a,wfkh1a,wfkb1a); plot(fkn1db,wfkh1db,wfkb1db);

2.3. Compare filters with different lengths

samplingF = 2048; //Hz samplingTime = 1; //seconds filterLength1 = 41; //points filterLength2 = 81; //points shape(x) = if x < (samplingF / 2) / 3 then 1 else 0; //say limit to a third of the interval fr(x)=sampleN(shape, 0, 1, samplingF/2+1); //freq. response //plot(fr); fk1(x)=ifft1(fr); //filter kernel 1 fk2(x)=cloneS(fk1); //filter kernel 2 //plot(fk); fkshr1(x)=shrotS(fk1,filterLength1/2); fktr1(x)=truncateS(fkshr1,0,filterLength1-1); fkn1(x)=fktr1(x)/sumS(fktr1); //normalized kernel 1 fkshr2(x)=shrotS(fk2,filterLength2/2); fktr2(x)=truncateS(fkshr2,0,filterLength2-1); fkn2(x)=fktr2(x)/sumS(fktr2); //normalized kernel 2 blackmanW(x,filterLength) = 0.42 - 0.5 * cos(2 * Pi * x / filterLength) + 0.08 * cos(4 * Pi * x / filterLength); //Blackman wfk1(x)=blackmanW(x/stepS(fkn1),filterLength1)*fkn1(x); //the Blackman windowed filter 1 wfk2(x)=blackmanW(x/stepS(fkn2),filterLength2)*fkn2(x); //the Blackman windowed filter 2 plot(wfk1,wfk2); //check the resulting filters frequency response efr1(x)=firResp(wfk1); //estimated freq. response length 1 efr2(x)=firResp(wfk2); //estimated freq. response length 2 plot(efr1,efr2,-.1,1.1,Pi+.1,-.5); db(x)=8.685890*log(x); efrdb1(x)=db(efr1(x)); //same as above but in db 1 efrdb2(x)=db(efr2(x)); //same as above but in db 2 plot(efrdb1,efrdb2,-.1,1,Pi+.1,-70);

2.4. Filter inverse, fiter reverse

samplingF = 2048; //Hz samplingTime = 1; //seconds filterLength = 81; //points shape(x) = if x < (samplingF / 2) / 3 then 1 else 0; //say limit to a third of the interval fr(x)=sampleN(shape, 0, 1, samplingF/2+1); //freq. response //plot(fr); fk(x)=ifft1(fr); //filter kernel 1 //plot(fk); fkshr(x)=shrotS(fk,filterLength/2); fktr(x)=truncateS(fkshr,0,filterLength-1); fkn(x)=fktr(x)/sumS(fktr); //normalized kernel blackmanW(x,filterLength) = 0.42 - 0.5 * cos(2 * Pi * x / filterLength) + 0.08 * cos(4 * Pi * x / filterLength); //Blackman wfk(x)=blackmanW(x/stepS(fkn),filterLength)*fkn(x); //the Blackman windowed filter fkInv(x)=inverseFilter(wfk); //inverse //fkInv(x)=reverseFilter(wfk); //reverse plot(wfk,fkInv); //check the resulting filters frequency response efr(x)=firResp(wfk); //estimated freq. response efrInv(x)=firResp(fkInv); //estimated freq. response length 2 plot(efr,efrInv,-.1,1.1,Pi+.1,-.5);

2.5. Band-pass/Band-reject out of a High-Pass and a Low-Pass

samplingF = 2048; //Hz samplingTime = 1; //seconds filterLength = 81; //points shape(x) = if x < (samplingF / 2) / 3 then 1 else 0; //say limit to a third of the interval fr(x)=sampleN(shape, 0, 1, samplingF/2+1); //freq. response //plot(fr); fk(x)=ifft1(fr); //filter kernel 1 //plot(fk); fkshr(x)=shrotS(fk,filterLength/2); fktr(x)=truncateS(fkshr,0,filterLength-1); fkn(x)=fktr(x)/sumS(fktr); //normalized kernel blackmanW(x,filterLength) = 0.42 - 0.5 * cos(2 * Pi * x / filterLength) + 0.08 * cos(4 * Pi * x / filterLength); //Blackman db(x)=8.685890*log(x); wfk(x)=blackmanW(x/stepS(fkn),filterLength)*fkn(x); //the Blackman windowed filter fkInv(x)=reverseFilter(wfk); //make high pass out of the low pass plot(wfk,fkInv); //check the resulting filters frequency response efr(x)=firResp(wfk); //estimated freq. response efrInv(x)=firResp(fkInv); plot(efr,efrInv,-.1,1.1,Pi+.1,-.5); //rf(x)=conv(wfk,fkInv); //band-pass -- THIS DOESN'T WORK, LOTS OF NOISE !!! rf(x)=wfk(x)+fkInv(x); //result filter band-reject rfn(x)=rf(x)/sumS(rf); plot(rfn); //rffr(x)=firResp(rfn); //plot(rffr,-.1,1.1,Pi+.1,-.5); //check the response but doing fft on the kernel fknp(x)=padS(rf,2^nextpow2(samplingF*samplingTime)); //pad to the proper length fkn1f(x)=fft1(fknp); //take fft fkn1a(x)=abs(fkn1f(x))*samplingF/2; //take abs and normalize fkn1db(x)=db(fkn1a(x)); //calculate in db plot(fkn1a); plot(fkn1db); //Since the conv line above doesn't work we can get bandpass by inversion of bandreject rfInv(x)=inverseFilter(rfn); plot(rfInv); rfInvFr(x)=firResp(rfInv); plot(rfInvFr,-.1,1.1,Pi+.1,-.5);

2.6. Interactive test

samplingF = 2048; //Hz samplingTime = 1; //seconds controlLength = 81; //points controlVar(controlLength, 50); nextOdd(x) = if isNextIntEven(x) then nextInt(x) + 1 else nextInt(x); filterLength=nextOdd(controlLength); rotLength=filterLength/2; truncLength=filterLength-1; cutoffF = .5; //fraction of the Niquist interval controlVar(cutoffF,.5); shape(x) = if x < (samplingF / 2) * cutoffF then 1 else 0; fr(x)=sampleN(shape, 0, 1, samplingF/2+1); //freq. response //plot(fr); fk(x)=ifft1(fr); //filter kernel 1 //plot(fk); fkshr(x)=shrotS(fk,rotLength); fktr(x)=truncateS(fkshr,0,truncLength); fkn(x)=fktr(x)/sumS(fktr); //normalized kernel blackmanW(x,filterLength) = 0.42 - 0.5 * cos(2 * Pi * x / filterLength) + 0.08 * cos(4 * Pi * x / filterLength); //Blackman db(x)=8.685890*log(x); wfk(x)=blackmanW(x/stepS(fkn),filterLength)*fkn(x); //the Blackman windowed filter plot(wfk); //check the resulting filters frequency response efr(x)=firResp(wfk); //estimated freq. response plot(efr,-.1,1.1,Pi+.1,-.5);