

Figure Matlab filter frequency response

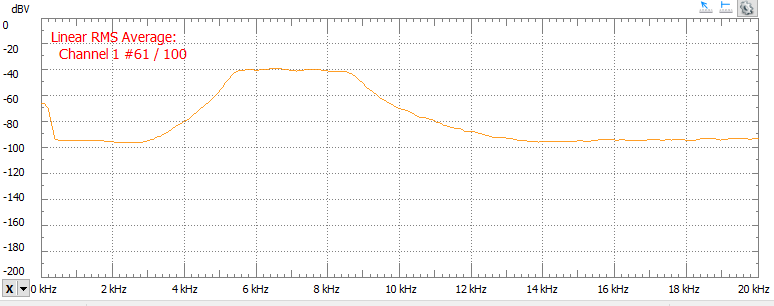


Figure 2 Cascade filter frequency response

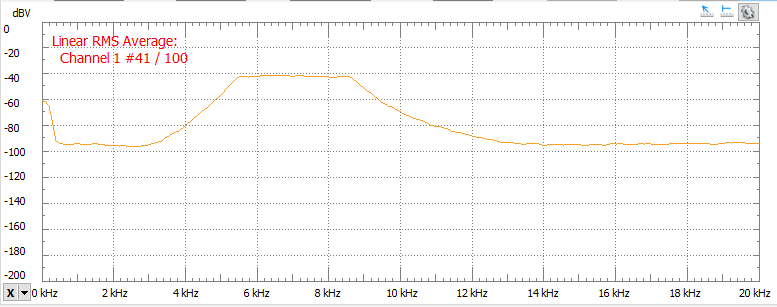


Figure Lattice-Ladder filter frequency response

Cascade.sa

.title "cascade.sa"

.def \_cascade

**.sect** ".cascade"

**.global** \_cascade

**.global** \_cascadeSection

**.global** \_filterSections

**.global** \_dBuff

**.global** \_dOffset

**.global** \_sections

**\_cascade:** .proc A4, B3 ;cascade(x(n))

.reg x, filter, dBuff, dOffset, sections, p\_filter, p\_dBuff, i, count, product, result, gain

;move globals into registers

mvk 0, i

mv A4, x

mvkl \_dBuff, dBuff

mvkh \_dBuff, dBuff

mvkl \_filterSections, filter

mvkh \_filterSections, filter

mvkl \_dOffset, dOffset

mvkh \_dOffset, dOffset

mvkl \_sections, sections

mvkh \_sections, sections

ldw \*dOffset, dOffset

ldw \*sections, sections

;loop over every section for cascade

**loop:**

mpy i, 4, count ;[i][4]

addaw dBuff, count, p\_dBuff ;&dBuff[i][0]

mpy i,7,count ;[i][7]

addaw filter, count, p\_filter ;&filterSections[i][0]

.call x = \_cascadeSection(x,p\_dBuff,dOffset,p\_filter) ;get output of single section

**add** i, 1, i

**sub** i, sections, count

[count] b loop

;apply output gain

mpy sections, 7, count ;[i][sections]

addaw filter, count, p\_filter ;obtain address of output gain

ldw \*p\_filter, gain ;load the output gain

mpysp x, gain, x

mv x, A4 ;return result

.endproc A4, B3

b B3

CascadeSection.sa

.title "cascadeSection.sa"

.def \_cascadeSection

**.sect** ".cascade"

**.global** \_cascadeSection

**\_cascadeSection:** .proc A4, B4, A6, B6 ;cascadeSection(x(n),\*dBuff(n=0),dOffset,filterCoef)

.reg x, dBuff, filter, p\_a, p\_b, a, b, d, product, dresult, yresult, count, dOffset, gain

mvkl 0x3<<16|0x1<<8, count

mvkh 0x3<<16|0x1<<8, count

mvc count,AMR ;make B4 a circular buffer of size 16(4\*wordSize)

;move parameters into local registers

mv A4, x

mv B6, filter

mv A6, dOffset

mvk 0,yresult

addaw filter, 4, p\_a ;a0 address 4\*wordLength

addaw filter, 1, p\_b ;b0 address 1\*wordLength

addaw B4, dOffset, B4 ;shift D to d(0) in circular buffer

mv B4, dBuff ;store initial d(0) location

;init d\_k to x\_k

ldw \*p\_a++, a ;load a0;

mpysp x,a, dresult ;x(n)\*a0

addaw p\_b,1,p\_b

mvk 2, count

;compute a,b\*d\_k

**dLoop:** ;i=1;i<3;i++

ldw \*p\_a++, a ;a(i)

ldw \*++B4, d ;d(n-i)

ldw \*p\_b++, b ;b(i)

mpysp d,a, product ;d(n-i)\*a(i)

subsp dresult,product,dresult

mpysp d,b,product ;d(n-i)\*b(i)

addsp product,yresult,yresult

**sub** count, 1, count

[count] b dLoop

;store d[0], calculate y+=d[0]\*b[0]

stw dresult, \*dBuff ;store d[0]

ldw \*+filter[1],b ;get b0

mpysp dresult, b, product ;d[0]\*b[0]

addsp product, yresult,yresult

;output gain

ldw \*filter, gain

mpysp yresult, gain, yresult ;y\*gain

mv yresult, A4 ;return y

.endproc A4, B3

b B3

LatticeLadder.sa

.title "latticeLadder.sa"

.def \_latticeLadder

**.sect** ".lattice"

**.global** \_latticeLadder

**.global** \_filterLength

**.global** \_kVal

**.global** \_vVal

**.global** \_gOld

**\_latticeLadder:** .proc A4,B3 ;latticeLadder(x(n))

.reg fVal,vNew, gNew, p\_gOld, gOld, p\_vBuff, vVal, p\_kBuff, kVal, i, m, pointer, product, output

;move into local registers

mv A4,fVal

mvkl \_filterLength, m

mvkh \_filterLength, m

ldw \*m,m

mvkl \_kVal, p\_kBuff

mvkh \_kVal, p\_kBuff

mvkl \_vVal, p\_vBuff

mvkh \_vVal, p\_vBuff

mvkl \_gOld, p\_gOld

mvkh \_gOld, p\_gOld

mvk 0,output

**sub** m,1,m ;filter\_size-1,for 0 index

**loop:**

;load array values

ldw \*+p\_kBuff[m], kVal ;k\_m

ldw \*+p\_vBuff[m], vVal ;v\_m

**sub** m,1,m ;used for g\_m-1

ldw \*+p\_gOld[m], gOld ;g\_m-1

;compute f\_m-1

mpysp kVal,gOld,product

subsp fVal,product,fVal ;f\_m-1=f\_m(n)-k\_m\*g\_m-1(n-1)

;compute g\_m

mpysp kVal,fVal,product

addsp product,gOld,gNew ;g\_m=k\_m\*f\_m-1(n)+g\_m-1(n-1)

;compute v\_m

mpysp vVal,gNew,product

addsp product,output,output ;y+=g\_m\*v\_m

;store gOld=gNew

**add** m,1,product

stw gNew,\*+p\_gOld[product] ;gOld[m+1]=gNew

[m] b loop

stw fVal,\*p\_gOld ;gOld[0]=f0

;compute final f\*v0+vOld

ldw \*p\_vBuff, vVal ;v0

mpysp vVal,fVal,product

addsp product,output, output ;output+=f0\*v0

mv output, A4

.endproc A4, B3

b B3