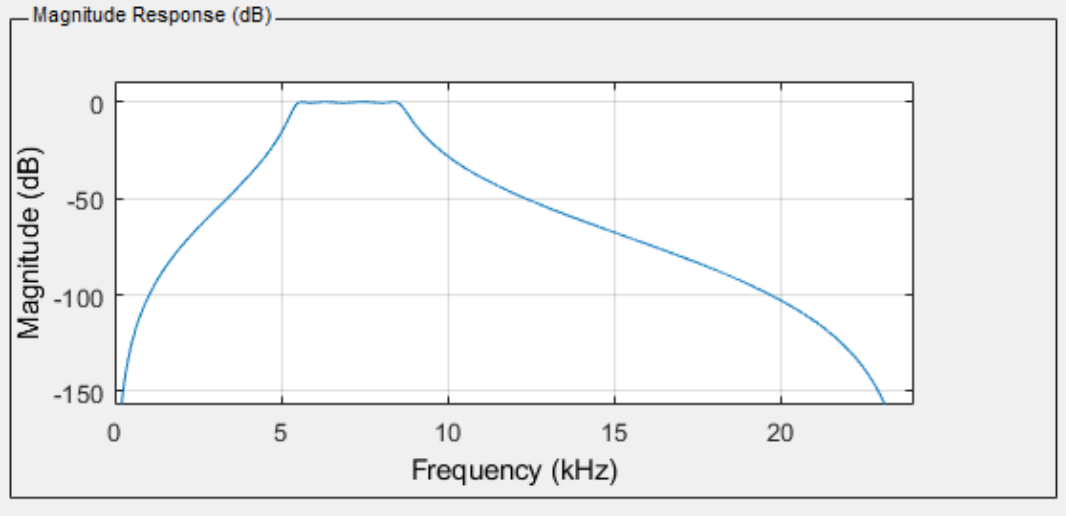


Figure 1 Matlab filter frequency response

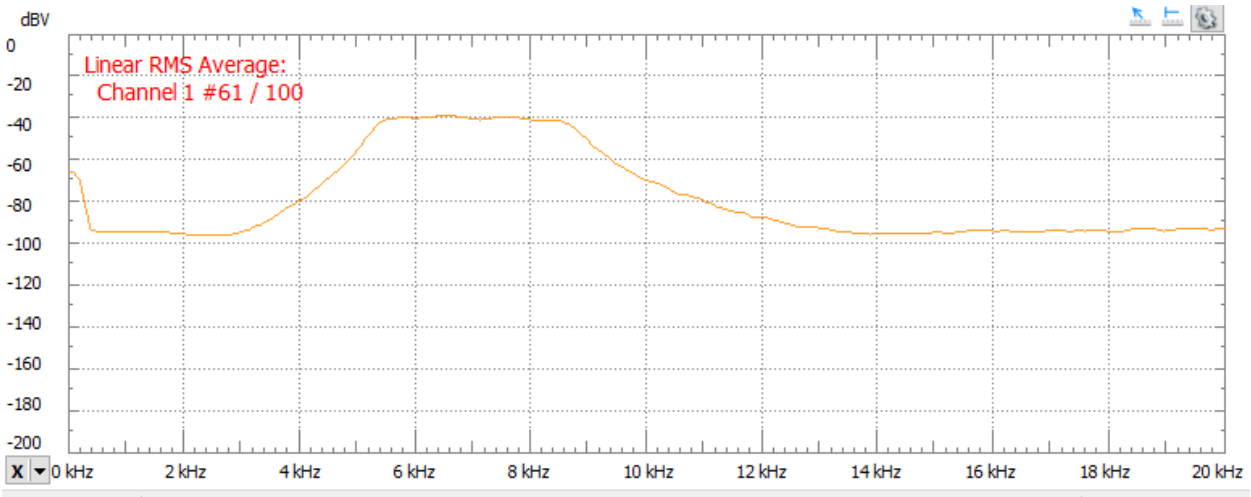


Figure 2 Cascade filter frequency response

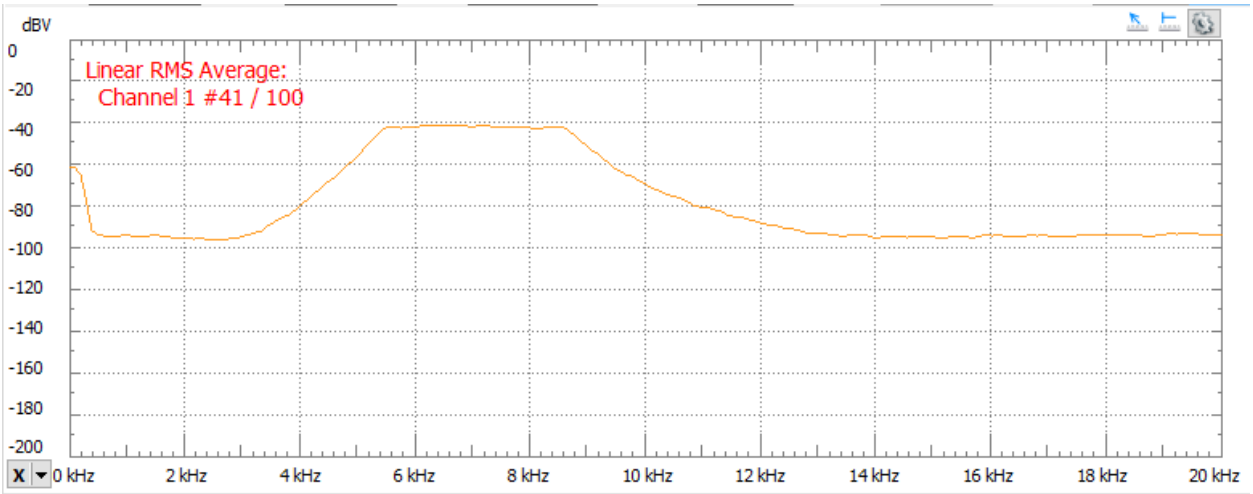


Figure 3 Lattice-Ladder filter frequency response

## 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
```