Scilab-Files for „signal.pdf“

```
//FLTS1.SCE
//filtering of signals
  xinit('flts1.ps')
  exec('Sflts1.sce')
  xend();
//SFLTS1.SCE
//filtering of signals
//make signal and filter
  [h,hm,fr] = wfir('lp',33,[.2 0],'hm',[0 0]);
  t = 1:200;
  x1 = sin(2*%pi*t/20);
  x2 = sin(2*%pi*t/3);
  x = x1+x2;
  plot(x);
//
```

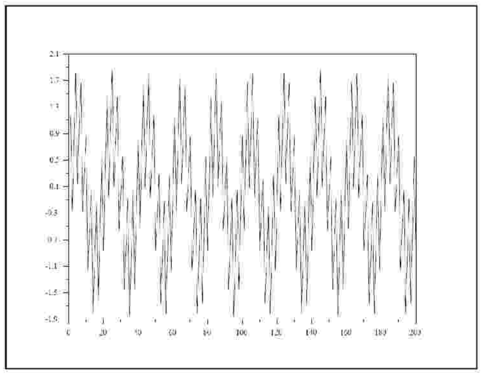

Figure 1.2: exec('flts1.code') Sum of Two Sinusoids

```
//FLTS2.SCE
//filtering with flts
  xinit('flts2.ps')
  exec('Sflts2.sce')
  xend()
//SFLTS2.SCE
//filtering of signals
//make signal and filter
  [h,hm,fr] = wfir('lp',33,[.2 0],'hm',[0 0]);
  t = 1:200;
  x1 = sin(2*%pi*t/20);
  x2 = sin(2*%pi*t/3);
  x = x1+x2;
  z = poly(0,'z');
  hz = syslin('d',poly(h,'z','c')./z**33);
  yhz = flts(x,hz);
  plot(yhz);
//
```

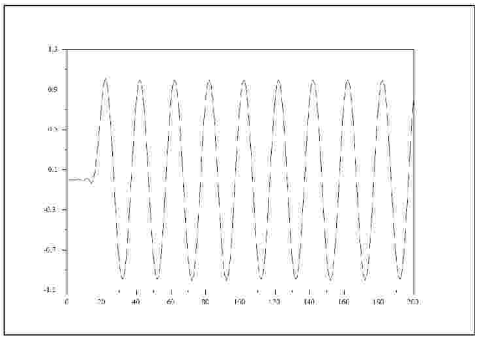

Figure 1.3: exec('flts2.code') Filtered Signal

```
//PLOT1.SCE
//Illustrate plot of FIR filter impulse response 'plot.1.sce'
  xinit('plot1.ps')
  exec('Splot1.sce')
  xend();
//SPLOT1.SCE
//Illustrate plot of FIR filter impulse response
  [h,hm,fr] = wfir('bp',55,[.2 .25],'hm',[0 0]);
  plot(h)
//
```

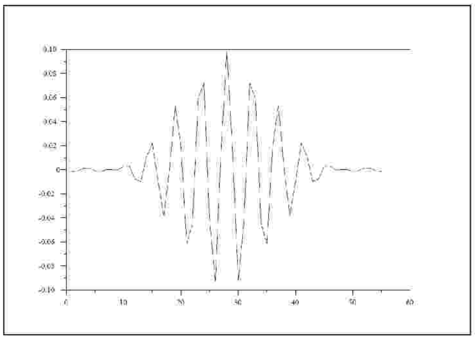

Figure 1.4: exec('plot1.code') Plot of Filter Impulse Response

```
//PLOT2.SCE
//Evaluate magnitude response of continuous-time system
  xinit('plot2.ps')
  exec('Splot2.sce')
  xend(),
//SPLOT2.SCE
//Evaluate magnitude response of continuous-time system
  hs = analpf(4,'cheb1',[.1 0],5)
  fr = 0:.1:15;
  hf = freq(hs(2),hs(3),%i*fr);
  hm = abs(hf);
  plot(fr,hm),
//
```

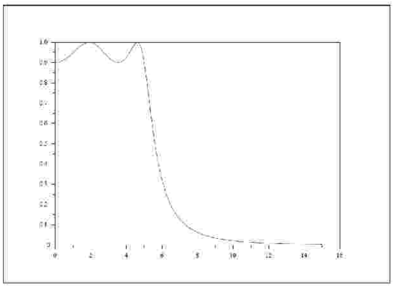

Figure 1.5: exec('plot2.code') Plot of Continuous Filter Magnitude Response

```
//PLOT3.SCE
//demonstrate Scilab function frmag
  xinit('plot3.ps')
  exec('Splot3.sce')
  xend(),
//SPLOT3.SCE
//demonstrate Scilab function frmag
  hn = eqfir(33,[0 .2;.25 .35;.4 .5],[0 1 0],[1 1 1]);
```

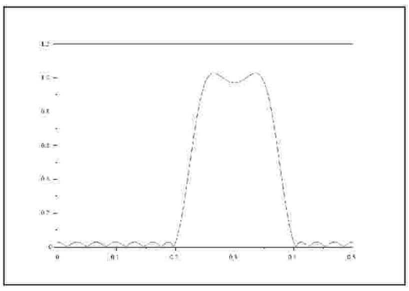

Figure 1.6: exec('plot3.code') Plot of Discrete Filter Magnitude Response

```
  [hm,fr] = frmag(hn,256);
  plot(fr,hm),
//
//PLOT4.SCE
//Demonstrate function plzr
  xinit('plot4.ps');
  exec('Splot4.sce')
  xend(),
//SPLOT4.SCE
//Demonstrate function plzr
  hz = iir(4,'lp','butt',[.25 0],[0 0])
  plzr(hz)
//
//BODE1.SCE
  xinit('bode1.ps')
  s = poly(0,'s');
  a = 10;
  h = real(s-a);
  ax = 0:.01:2;
  ax = exp(log(10)*ax);
  hm = 20*log(abs(freq(1,h,%i*ax)))/log(10);
  plot2d1("gln",ax',hm',[-2],"011",' ',[1,-40,100,-10],[10,2,10,2]);
  x = -20*log(a)/log(10);
//horizontal straight line approximation
  plot2d1("gln",[1 a]',[x x]',[-3],"000");
//-20 db/dec straight line approximation
  plot2d1("gln",[a 100]',[x -40]',[-1],"000");
//vertical line at -3db
  plot2d1("gln",[a a]',[-20 -23]',[-1],"000");
  xtitle(' ','Log scale',' ')
  xend()
//
//BODE2.SCE
  exec('bode1.sce');
//plot phase
  xinit('bode2.ps')
  h = a;
  g = s;
  ax = -1:.1:3;
  ax = exp(log(10)*ax);
  hm = freq(h,1,ax);
  gm = freq(g,1,ax);
  gh = gm./hm;
//pm = pm+ofst;
  pm = -atan(real(gh));
  plot2d1("gln",ax',360*pm'/(2*%pi),[-2],"011",' ',[1,-90,10^3,0]);
  plot2d1("gln",[.1 1000]',[-45 -45]',[-3],"000");
  plot2d1("gln",[10 10]',[0 -180]',[-3],"000");
  xend()
//
//BODE3.SCE
  xinit('bode3.ps');
  s = poly(0,'s');
  a = 10;
  b = 25;
  h = real((s-(a-%i*b))*(s-(a+%i*b)));
  ax = 0:.01:2;
  ax = exp(log(10)*ax);
  hm = 20*log(abs(freq(1,h,%i*ax)))/log(10);
  plot2d1("gln",ax',hm',[-2],"011",' ',[1,-80,100,-50],[5,2,5,2]);
  x = -20*log(a**2+b**2)/log(10);
  y = sqrt(b**2-a**2);
  z = sqrt(b**2+a**2);
//horizontal straight line approximation
  plot2d1("gln",[1 z]',[x x]',[-1],"000");
//-40 db/dec straight line approximation
```

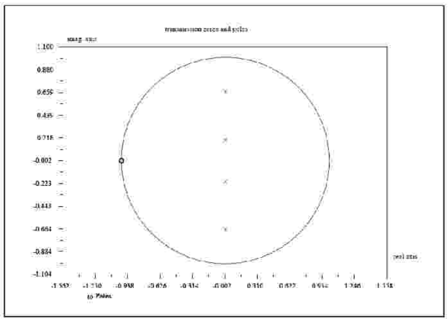

Figure 1.7: exec('plot4.code') Plot of Poles and Zeros of IIR Filter

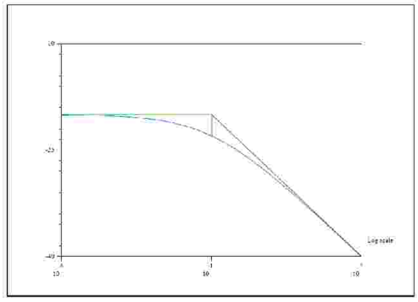

Figure 2.1: exec('bode1.code') Log-Magnitude Plot of $H(s) = 1/(s-a)$

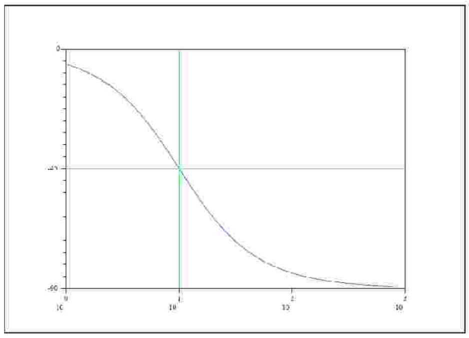

Figure 2.2: exec('bode2.code') Phase Plot of $H(s) = 1/(s-a)$

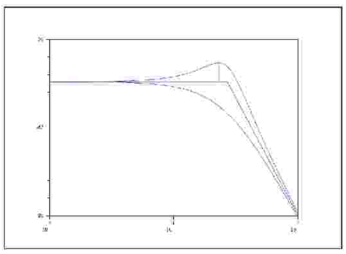

Figure 2.3: exec('bode3.code') Log-Magnitude Plot of $H(s) = (s^2 - 2as + (a^2 + b^2))^{-1}$

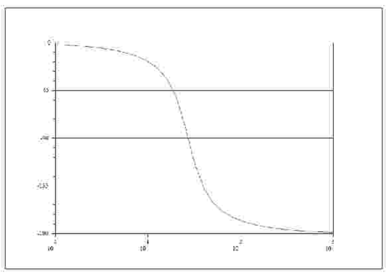

Figure 2.4: exec('bode4.code') Phase Plot of $H(s) = (s^2 - 2as + (a^2 + b^2))^{-1}$

```
  plot2d1("gln",[z 100]',[x -80]',[-1],"000");
//vertical line showing sqrt(b*b-a*a)
  plot2d1("gln",[y y]',[x -20*log(2*a*b)/log(10)]',[-1],"000");
  a = 25;
  b = 10;
  h = real((s-(a-%i*b))*(s-(a+%i*b)));
  hm = 20*log(abs(freq(1,h,%i*ax)))/log(10);
  plot2d1("gln",ax',hm',[-2],"000");
  xend()
//________________________________________________________________
//BODE4.SCE
  exec('bode3.sce')
  xinit('bode4.ps');
//plot phase
  h = 725-s*s;
  g = -20*s;
  ax = 0:.1:3;
  ax = exp(log(10)*ax);
  hm = freq(h,1,ax);
  gm = freq(g,1,ax);
  gh = gm./hm;
  pm = -(log(ones(gh)-%i*gh)-log(ones(gh)+%i*gh))/(2*%i);
  ofst = %pi*(-ones(hm)+hm./abs(hm))/2;
  pm = real(pm+ofst);
  plot2d1("gln",[1 1 1000]',[0 -180 -180]',[-3],"000");
  plot2d1("gln",ax',360*pm'/(2*%pi),[-2],"011",' ',[1,-180,1000,0],[5,4,5,4]);
  plot2d1("gln",[1 1000]',[-45 -45]',[-1],"000");
  plot2d1("gln",[1 1000]',[-90 -90]',[-1],"000");
  plot2d1("gln",[1 1000]',[-135 -135]',[-1],"000");
  plot2d1("gln",[26.9 26.9]',[0 -180]',[-1],"000");
  xend()
//________________________________________________________________
//BODE5.SCE
  xinit('bode5.ps')
  exec('Sbode5.sce')
  xend();
//SBODE5.SCE
//Bode plot
  a = -2*%pi;b = 1;c = 18*%pi;d = 1;
  sl = syslin('c',a,b,c,d);
  bode(sl,.1,100);
//________________________________________________________________
//BODE6.SCE
  xinit('bode6.ps')
  exec('Sbode6.sce')
  xend();
//SBODE6.SCE
//Bode plot; rational polynomial
  s = poly(0,'s');
  h1 = 1/real((s+2*%pi*(15+100*%i))*(s+2*%pi*(15-100*%i)))
  h1 = syslin('c',h1);
  bode(h1,10,1000,.01);
//________________________________________________________________
//BODE7.SCE
  xinit('bode7.ps');
  exec('Sbode7.sce')
  xend(),
//SBODE7.SCE
//Bode plot; two systems in series
  a = -2*%pi;b = 1;c = 18*%pi;d = 1;
  sl = syslin('c',a,b,c,d);
  s = poly(0,'s');
  h1 = 1/real((s+2*%pi*(15+100*%i))*(s+2*%pi*(15-100*%i)));
  h1 = syslin('c',h1);
  h2 = ss2tf(sl)
  bode(h1*h2,.1,1000,.01);
```

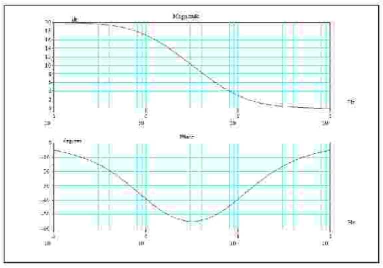

Figure 2.5: exec('bode5.code') Bode Plot of State-Space System Representation

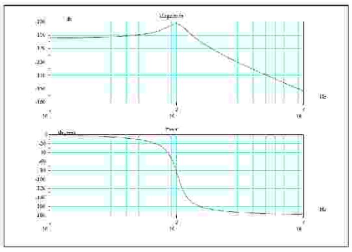

Figure 2.6: exec('bode6.code') Bode Plot of Rational Polynomial System Representation

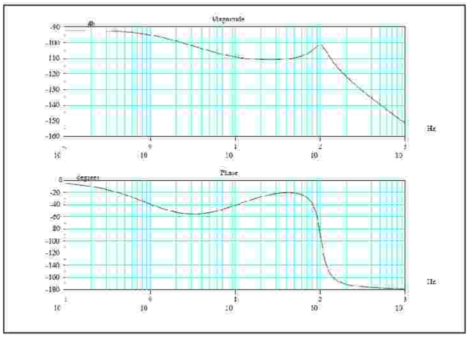

Figure 2.7: exec('bode7.code') Bode Plot Combined Systems

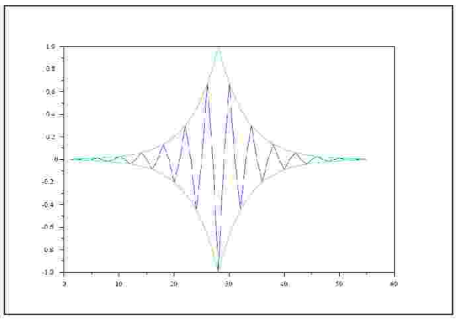

Figure 2.8: exec('group1_5.code') Modulated Exponential Signal

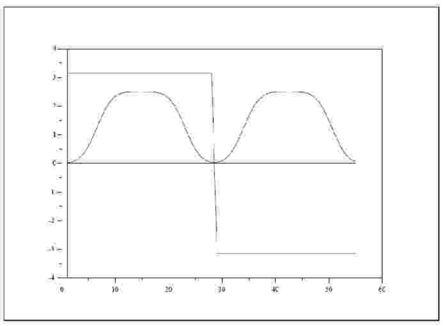

Figure 2.9: exec('group1_5.code') Constant Phase Band Pass Filter

```
//______________________________________________________________________

//GROUP1_5.SCE
//create carrier and narrow band signal
  xinit('group1.ps');
  wc = 1/4;
  x = sin(2*%pi*(0:54)*wc);
  y = exp(-abs(-27:27)/5);
  f = x.*y;
  plot([1 1 55],[1 -1 -1]),
  nn = prod(size(f))
  plot2d((1:nn)',f',[-2],"000"),
  nn = prod(size(y))
  plot2d((1:nn)',y',[-3],"000"),
  plot2d((1:nn)',-y',[-3],"000"),
  xend(),
  xinit('group2.ps');
//make band pass filter
  [h w] = wfir('bp',55,[maxi([wc-.15,0]),mini([wc+.15,.5])],'kr',60.);
//create new phase function with only phase delay
  hf = fft(h,-1);
  hm = abs(hf);
  hp = %pi*ones(1:28);//tg is zero
  hp(29:55) = -hp(28:-1:2);
  hr = hm.*cos(hp);
  hi = hm.*sin(hp);
  hn = hr+%i*hi;
  plot([1 1 55],[4 -4 -4]),
  plot2d([1 55]',[0 0]',[-1],"000"),
  nn = prod(size(hp))
  plot2d((1:nn)',hp',[-2],"000"),
  nn = prod(size(hm))
  plot2d((1:nn)',2.5*hm',[-1],"000"),
  xend(),
  xinit('group3.ps');
//filter signal with band pass filter
  ff = fft(f,-1);
  gf = hn.*ff;
  g = fft(gf,1);
  plot([1 1 55],[1 -1 -1]),
  nn = prod(size(g))
  plot2d((1:nn)',real(g)',[-2],"000"),
  nn = prod(size(f))
  plot2d((1:nn)',f',[-1],"000"),
  xend(),
//create new phase function with only group delay
  xinit('group4.ps');
  tg = -1;
  hp = tg*(0:27)-tg*12.*ones(1:28)/abs(tg);//tp is zero
  hp(29:55) = -hp(28:-1:2);
  hr = hm.*cos(hp);
  hi = hm.*sin(hp);
  hn = hr+%i*hi;
  plot([1 1 55],[15 -15 -15]),
  plot2d([1 55]',[0 0]',[-1],"000"),
  nn = prod(size(hp))
  plot2d((1:nn)',hp',[-2],"000"),
  nn = prod(size(hm))
  plot2d((1:nn)',10*hm',[-1],"000"),
  xend(),
  xinit('group5.ps');
//filter signal with band pass filter
  ff = fft(f,-1);
  gf = hn.*ff;
  g = fft(gf,1);
  plot([1 1 55],[1 -1 -1]),
```

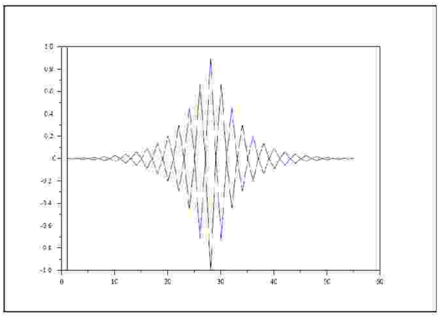

Figure 2.10: exec('group1_5.code') Carrier Phase Shift by $t_p = \pi/2$

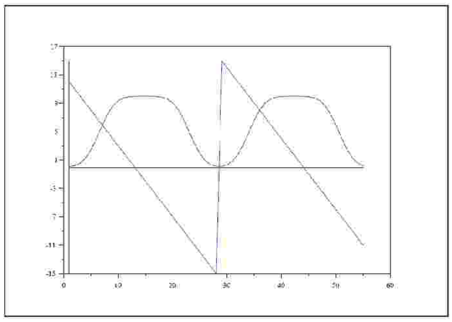

Figure 2.11: exec('group1_5.code') Linear Phase Band Pass Filter

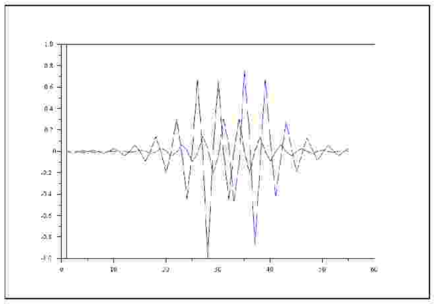

Figure 2.12: exec('group1_5.code') Envelope Phase Shift by $t_g = -1$

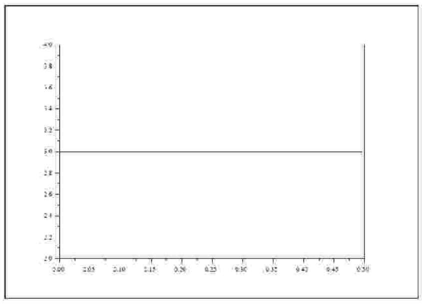

Figure 2.13: exec('group6_8.code') Group Delay of Linear-Phase Filter

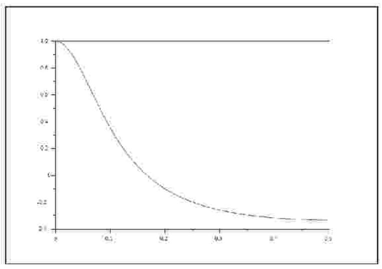

Figure 2.14: exec('group6_8.code') Group Delay of Filter (Rational Polynomial)

```
 nn = prod(size(g))
 plot2d((1:nn)',real(g)',[-2],"000"),
 nn = prod(size(f))
 plot2d((1:nn)',f',[-1],"000"),
 xend(),
//
//GROUP6_8.SCE
//save and change dess
 xinit('group6.ps');
//create filter using ffir
 [h w] = wfir('lp',7,[.2,0],'hm',[0.01,-1]);
 [tg,fr] = group(100,h);
 plot2d(fr',tg',1,'011',' ',[0,2,0.5,4.])
 xend()
//demonstrate rational polynomial
 xinit('group7.ps');
 z = poly(0,'z');
 h = z/(z-.5);
 [tg,fr] = group(100,h);
 plot(fr,tg)
 xend()
//demonstrate cascade realization
 xinit('group8.ps');
 h = [1 1.5 -1 1;2 -2.5 -1.7 0;3 3.5 2 5];
 h = h';
 h = casc(h,'z');
 [tg,fr] = group(100,h);
 plot(fr,tg)
 xend()
//
//SAMPLE1.SCE
 xinit('sample1.ps')
//no axes or tick marks
//plot axes
 rect = [-5,-2,5,5]
 plot2d([-.05 0 .05]',[4.8 5 4.8]',[-1],"011",' ',rect),
 plot2d([-5 5]',[0 0]',[-1],"000"),
 plot2d([0 0]',[-2 5]',[-1],"000"),
 plot2d([4.8 5 4.8]',[.05 0 -.05]',[-1],"000"),
//plot figure
 plot2d([-1 0 1]',[0 2 0]',[-1],"000")
 xstring(1,0,'Wc'),
 xstring(-1,0,'-Wc'),
 xstring(0,4.5,'X(W)')
 xstring(4.5,0,'W')
 xstring(0,2,'X(0)'),
 xend(),
//
//SAMPLE2.SCE
 xinit('sample2.ps')
//plot axes
 rect = [-5,-2,5,5]
 plot2d([-.05 0 .05]',[4.8 5 4.8]',[-1],"011",' ',rect),
 plot2d([-5 5]',[0 0]',[-1],"000"),
 plot2d([0 0]',[-2 5]',[-1],"000"),
 plot2d([4.8 5 4.8]',[.05 0 -.05]',[-1],"000"),
//plot figure
 plot2d([-1 0 1]',[0 2 0]',[-1],"000")
 plot2d([-3.5 -2.5 -1.5]',[0 2 0]',[-1],"000")
 plot2d([1.5 2.5 3.5]',[0 2 0]',[-1],"000")
 plot2d([1.25 1.25]',[-.1 .1]',[-1],"000")
 plot2d([-1.25 -1.25]',[-.1 .1]',[-1],"000")
 plot2d([2.5 2.5]',[-.1 .1]',[-1],"000")
 plot2d([-2.5 -2.5]',[-.1 .1]',[-1],"000")
 xstring(1,-.5,'pi/T'),
 xstring(0,4.5,'X(W)')
```

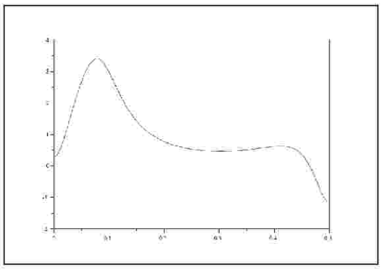

Figure 2.15: exec('group6_8.code') Group Delay of Filter (Cascade Realization)

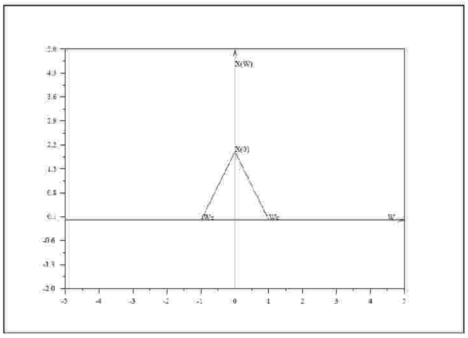

Figure 2.16: exec('sample1.code') Frequency Response $X(\Omega)$

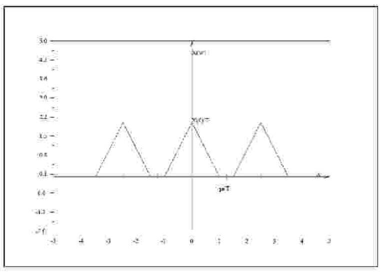

Figure 2.17: exec('sample2.code') Frequency Response $x(\omega)$ With No Aliasing

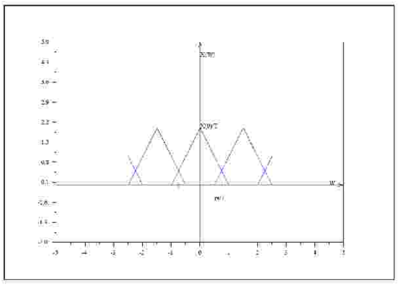

Figure 2.18: exec('sample3.code') Frequency Response $x(\omega)$ With Aliasing

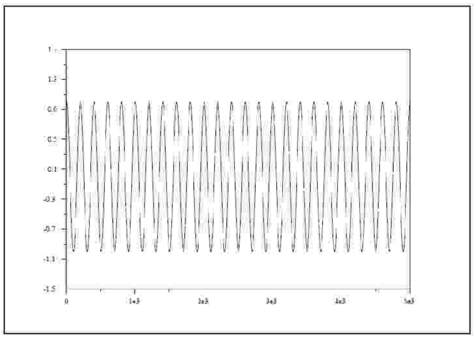

Figure 2.19: exec('sample4.code') Cosine Signal

```
 xstring(4.5,0,'W')
 xstring(0,2,'X(0)/T')
 xend(),
//
//SAMPLE3.SCE
 xinit('sample3.ps')
 rect = [-5,-2,5,5]
 plot2d([-.05 0 .05]',[4.8 5 4.8]',[-1],"011",' ',rect),
 plot2d([-5 5]',[0 0]',[-1],"000"),
 plot2d([0 0]',[-2 5]',[-1],"000"),
 plot2d([4.8 5 4.8]',[.05 0 -.05]',[-1],"000"),
//plot figure
 plot2d([-.5 0 .5]',[1 2 1]',[-1],"000"),
 plot2d([-2.5 -2 -1.5 -1 -.5]',[1 1 2 1 1]',[-1],"000"),
 plot2d([.5 1 1.5 2 2.5]',[1 1 2 1 1]',[-1],"000"),
 plot2d([-1 -.5]',[0 1]',[-2],"000"),
 plot2d([-1 -.5]',[1 0]',[-2],"000"),
 plot2d([.5 1]',[0 1]',[-2],"000"),
 plot2d([.5 1]',[1 0]',[-2],"000"),
 plot2d([-2.5 -2]',[0 1]',[-2],"000"),
 plot2d([-2.5 -2]',[1 0]',[-2],"000"),
 plot2d([2 2.5]',[0 1]',[-2],"000"),
 plot2d([2 2.5]',[1 0]',[-2],"000"),
 plot2d([.75 .75]',[-.1 .1]',[-2],"000"),
 plot2d([-.75 -.75]',[-.1 .1]',[-2],"000"),
 xstring(0.5,-.5,'pi/T'),
 xstring(0,4.5,'X(W)')
 xstring(4.5,0,'W')
 xstring(0,2,'X(0)/T')
 xend(),
//
//SAMPLE4.SCE
 xinit('sample4.ps')
 x = cos(2*%pi*(0:4999)/200);
 y = 0*ones(x);
 y = x(1:105:5000);
 yn = cos(2*%pi*(0:47)/40);
 plot([0 0 5000],[1.5 -1.5 -1.5]),
 n = prod(size(x))
 plot2d((1:n)',x',[-1],"000"),
 xend(),
//
//SAMPLE4_5.SCE
 x = cos(2*%pi*(0:4999)/200);
 y = 0*ones(x);
 y = x(1:105:5000);
 yn = cos(2*%pi*(0:47)/40);
 plot([0 0 5000],[1.5 -1.5 -1.5]),
 n = prod(size(x))
 plot2d((1:n)',x',[-1],"000"),
 xend(),
 plot([0 0 48],[1.5 -1.5 -1.5]);
 n = prod(size(y))
 plot2d((1:n)',y',[-1],"000"),
 n = prod(size(yn))
 plot2d((1:n)',yn',[-2],"000"),
 plot2d((1:n)',-yn',[-2],"000"),
 xend(),
//
//SAMPLE5.SCE
 exec('sample4.sce')
 xinit('sample5.ps')
 plot([0 0 48],[1.5 -1.5 -1.5]);
 n = prod(size(y))
 plot2d((1:n)',y',[-1],"000"),
 n = prod(size(yn))
```

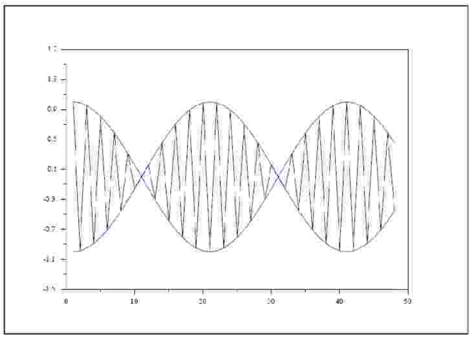

Figure 2.20: exec('sample5.code') Aliased Cosine Signal

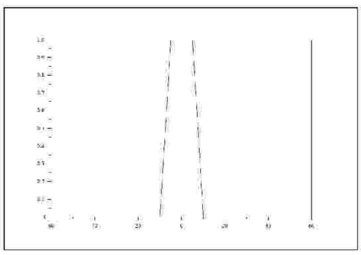

Figure 2.21: exec('intdec1_4.code') Fourier Transform of a Continuous Time Signal

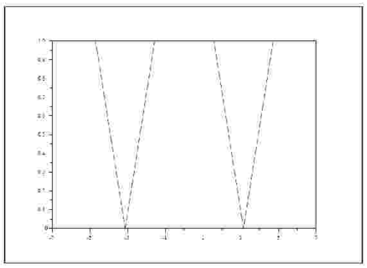

Figure 2.22: exec('intdec1_4.code') Fourier Transform of the Discrete Time Signal

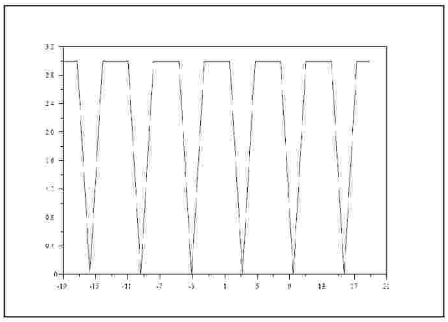

Figure 2.23: exec('intdec1_4.code') Fourier Transform of $v(nT')$

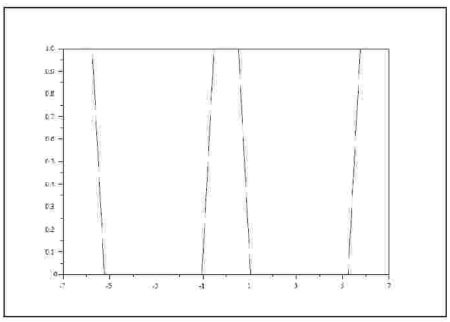

Figure 2.24: exec('intdec1_4.code') Fourier Transform of $x(nT')$

```
 plot2d((1:n)',yn',[-2],"000"),
 plot2d((1:n)',-yn',[-2],"000"),
 xend(),
//_______________________________________________________
//INTDEC1_4.SCE
 xinit('intdec1.ps');
 axis = (-60:60);
 f(1:5) = (0:4)/5;
 f(6:16) = ones(1:11);
 f(17:20) = (4:-1:1)/5;
 x(1:121) = 0*ones(1:121);
 x(51:70) = f(1:20);
 fs(1:15) = (0:14)/15;
 fs(16:46) = ones(1:31);
 fs(47:60) = (14:-1:1)/15;
 xs(1:30) = fs(31:60);
 xs(31:90) = fs(1:60);
 xs(91:121) = fs(1:31);
 v(1:10) = f(11:20);
 v(11:30) = f(1:20);
 v(31:50) = f(1:20);
 v(51:70) = f(1:20);
 v(71:90) = f(1:20);
 v(91:110) = f(1:20);
 v(111:121) = f(1:11);
 v = v*3;
 xh = 0*ones(1:121);
 xh(1:10) = f(11:20);
 xh(51:70) = f(1:20);
 xh(111:121) = f(1:11);
 plot(axis,x)
 xend()
 xinit('intdec2.ps');
 plot(2*%pi*axis/60,xs)
 xend()
 xinit('intdec3.ps');
 plot(6*%pi*axis/60,v)
 xend()
 xinit('intdec4.ps');
 plot(2*%pi*axis/60,xh)
 xend()
//_______________________________________________________
//INTDEC5_10.SCE
 xinit('intdec5.ps')
 xf = 24:-1:0;
 xf(26:50) = xf(25:-1:1);
 xs = real(fft(xf,1));
 x = xs(26:50);
 x(26:50) = xs(1:25);
 l = 8;
 m = 5;
 forder = 33;
//Get dimensions of vectors
 hsize = forder;
 xsize = maxi(size(x));
 xlsize = (xsize-1)*l+1;
 xhsize = xlsize+hsize;
//Design FIR low-pass filter with cut-off frequency fco = min(fr/2,l*fr/(2*m))
 fr = .5*xsize/(xhsize-1);
 fco = mini([fr,l*fr/m]);
 [hffir,w] = wfir('lp',forder,[fco,0],'hm',[0.01,-1]);
 h = l*hffir';
//upsample x by putting l-1 zeroes between each sample
 xl(1:l:xlsize) = x;
//prepare xl for linear convolution
 xl(xlsize+1:xhsize-1) = 0*ones(1:hsize-1);
```

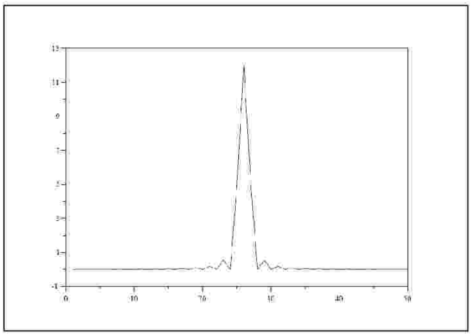

Figure 2.26: exec('intdec5_10.code') The Sequence $x(nT)$

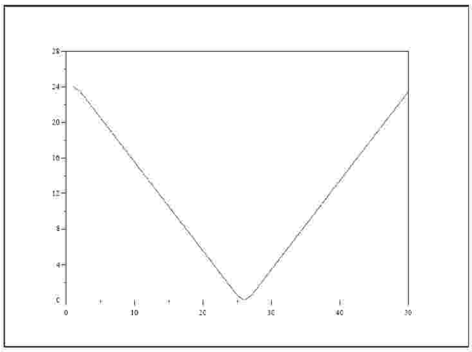

Figure 2.27: exec('intdec5_10.code') The DFT of $x(nT)$

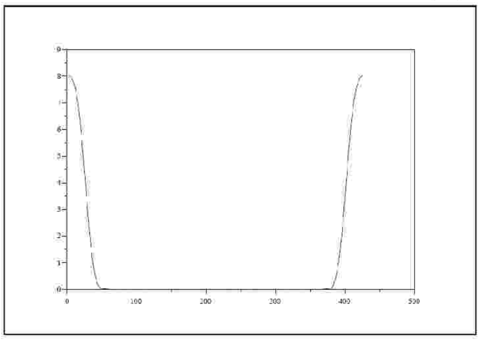

Figure 2.28: exec('intdec5_10.code') Low Pass Filter

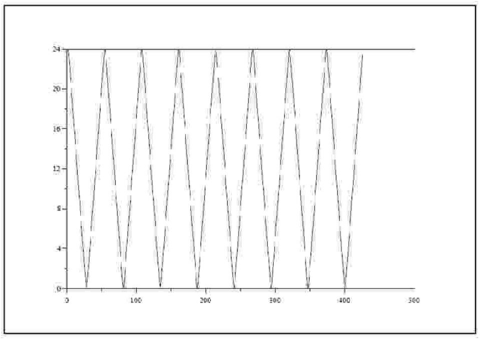

Figure 2.29: exec('intdec5_10.code') DFT of $v(nT)$

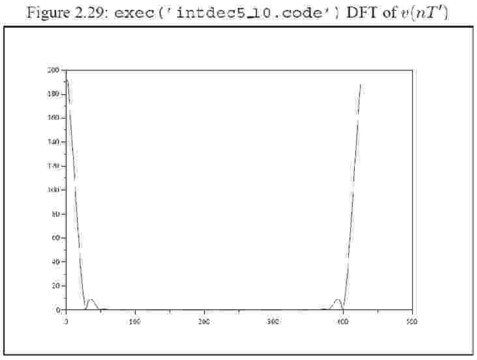

Figure 2.30: exec('intdec5_10.code') Filtered Version of V

```
//prepare h for linear convolution
  hl = 0*ones(xl);
  hl(1:hsize) = h';
//circular convolution by fft
  hlf = fft(hl,-1);
  xlf = fft(xl,-1);
  yl = real(fft(hlf.*xlf,1));
//remove leading and trailing elements due to partial convolution with h
  hso2 = int(hsize/2);
  ylmod = yl(hso2+1:xhsize-hso2-1);
//downsample ylmod by taking every mth sample
  y = ylmod(1:m:xhsize-2*hso2-1);
  plot(x)
  xend()
  xinit('intdec6.ps')
  plot(abs(fft(x,-1)))
  xend()
  xinit('intdec7.ps')
  plot(abs(hlf))
  xend()
  xinit('intdec8.ps')
  plot(abs(xlf))
  xend()
  xinit('intdec9.ps')
  plot(abs(xlf.*hlf))
  xend()
  xinit('intdec10.ps')
  plot(y)
  xend()
//______________________________________________________________
//FFT1.SCE
  xinit('fft1.ps')
  exec('Sfft1.sce')
  xend(),
//SFFT1.SCE
//Data for fft1
  x = 0:63;y = cos(2*%pi*x/16);
  yf = fft(y,-1);
  plot(x,y);
//______________________________________________________________
//FFT1_2.SCE
  xinit('fft1.ps')
  x = 0:63;
  y = cos(2*%pi*x/16);
  yf = fft(y,-1);
  plot(x,y);
  xend(),
//______________________________________________________________
//FFT2.SCE
  xinit('fft2.ps')
  exec('Sfft2.sce');
  plot(x,real(yf)');
  xend(),
//SFFT2.SCE
//Simple use of fft
  x = 0:63;y = cos(2*%pi*x/16);
  yf = fft(y,-1);
  plot(x,real(yf)');
  xend(),
//______________________________________________________________
//CZT1.SCE
  xinit('czt1.ps');
  a = .7*exp(%i*%pi/6);
  rect = [-1.2,-1.2*sqrt(2),1.2,1.2*sqrt(2)];
  t = 2*%pi*(0:179)/179;
  xsetech([0,0,0.5,1]);
```

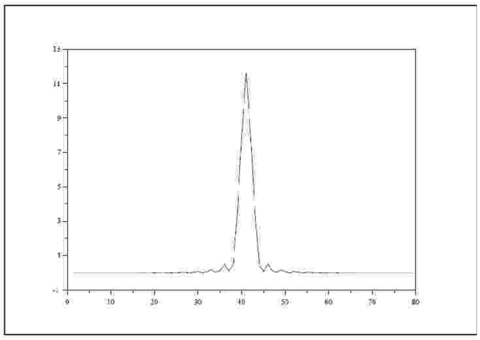

Figure 2.31: exec('intdec5_10.code') Sequence $x(nMT/L)$

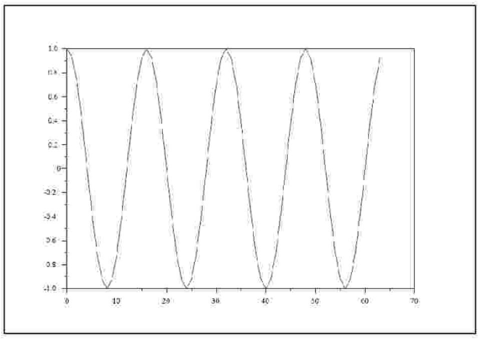

Figure 2.32: exec('fft1.code') Cosine Signal

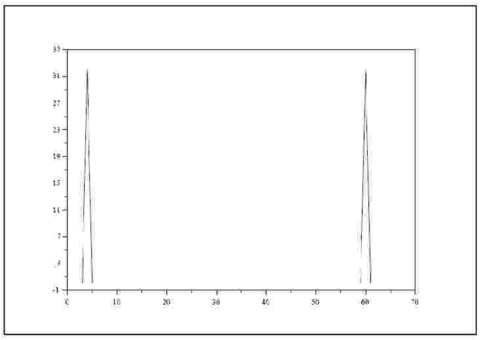

Figure 2.33: exec('fft2.code') DFT of Cosine Signal

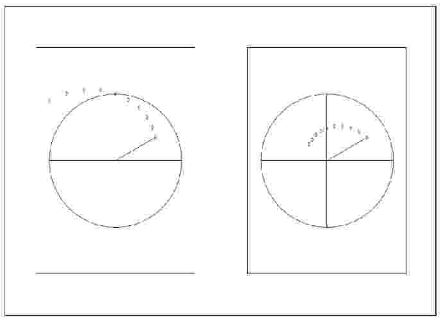

Figure 2.35: exec('czt1.code') Samples of the z-transform on Spirals

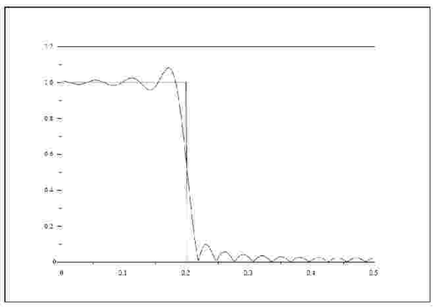

Figure 3.1: exec('fir1.code') Rectangularly windowed low-pass filter

```
plot2d(sin(t)',cos(t)',[-1],"012",' ',rect)
plot2d([0 real(a)]',[0 imag(a)]',[-1],"000")
xsegs([-1.0,0;1.0,0],[0,-1.0;0,1.0])
w0 = .93*exp(-%i*%pi/15);
w = exp(-(0:9)*log(w0));
z = a*w;
zr = real(z);
zi = imag(z);
plot2d(zr',zi',[5],"000")
xsetech([0.5,0,0.5,1]);
plot2d(sin(t)',cos(t)',[-1],"012",' ',rect)
plot2d([0 real(a)]',[0 imag(a)]',[-1],"000")
xsegs([-1.0,0;1.0,0],[0,-1.0;0,1.0])
w0 = w0/(.93*.93);
w = exp(-(0:9)*log(w0));
z = a*w;
zr = real(z);
zi = imag(z);
plot2d(zr',zi',[5],"000")
xend()
//
//FIR1.SCE
xinit('fir1.ps')
[wft,wfm,fr] = wfir('lp',33,[.2 0],'re',[0 0]);
hdes = ones(1:102);
hdes(103:256) = 0*ones(1:154);
plot(fr,wfm)
plot2d(fr',hdes',[-1],"000")
//
//FIR2_5.SCE
xinit('fir2.ps')
w = %pi*(-1:2/200:1);
lp = 0*ones(w);
hp = lp;
bp = lp;
sb = lp;
for j = 1:201,if abs(w(j))< = %pi/4 then lp(j) = 1;end;end
for j = 1:201,if abs(w(j))> = 3*%pi/4 then hp(j) = 1;end;end;
for j = 1:201,if abs(w(j)-3*%pi/8)< = %pi/8 then bp(j) = 1;end;end
for j = 1:201,if abs(w(j)+3*%pi/8)< = %pi/8 then bp(j) = 1;end;end;
for j = 1:201,if abs(w(j))< = %pi/4 then sb(j) = 1;end;end
for j = 1:201,if abs(w(j))< = %pi/2 then sb(j) = 1;end;end;
plot2d(w',lp',[-1],"011"," ",[-%pi,-0.125,%pi,1.125])
xend()
xinit('fir3.ps')
plot2d(w',hp',[-1],"011"," ",[-%pi,-0.125,%pi,1.125])
xend()
xinit('fir4.ps')
plot2d(w',bp',[-1],"011"," ",[-%pi,-0.125,%pi,1.125])
xend()
xinit('fir5.ps')
plot2d(w',sb',[-1],"011"," ",[-%pi,-0.125,%pi,1.125])
xend()
//
//FIR6.SCE
xinit('fir6.ps')
n = 33;
w = 8*%pi*(-1:.01:1)/n;
rn = sin(w*n/2);
rn(101) = n;
rd = sin(w/2);
rd(101) = 1;
rw = abs(rn./rd);
dess(31) = 1;
plot(w,rw)
plot2d([-2*%pi/n 0 2*%pi/n]',[0 n 0]',[-3],"000")
```

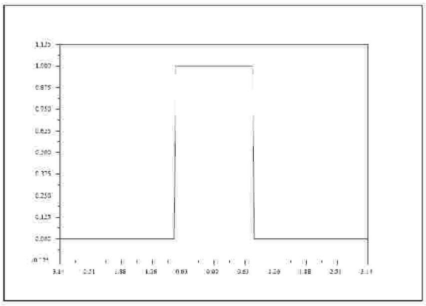

Figure 3.2: exec('fir2_5.code') Frequency response of a low pass filter

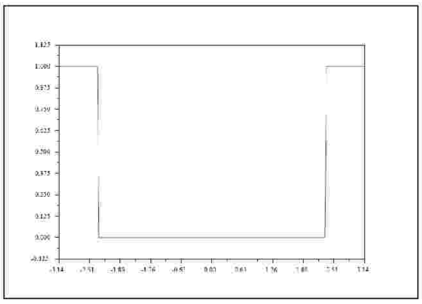

Figure 3.3: exec('fir2_5.code') Frequency response of a high pass filter

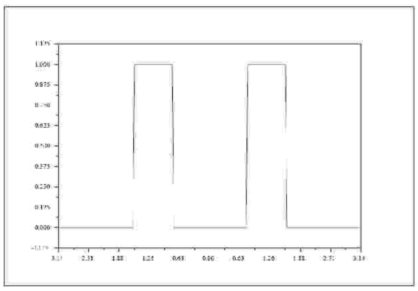

Figure 3.4: exec('fir2_5.code') Frequency response of a band pass filter

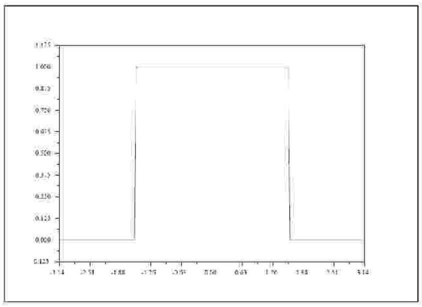

Figure 3.5: exec('fir2_5.code') Frequency response of a stop band filter

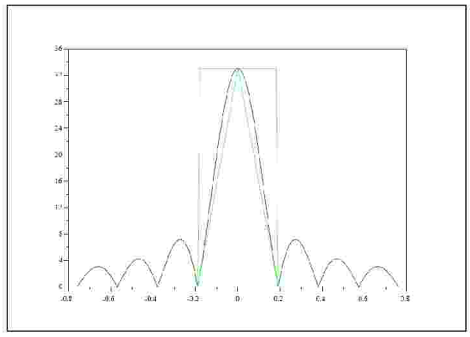

Figure 3.6. exec('fir6.code') Magnitude of rectangular window

```
plot2d([-2*%pi/n .01-2*%pi/n -.01+2*%pi/n 2*%pi/n]',[0 n n 0]',[-3],"000")
xend(),
//_____________________________________________
//FIR7.SCE
xinit('fir7.ps')
[wft,wfm,fr] = wfir('lp',33,[.2 0],'kr',[5.6 0]);
plot(fr,log(wfm))
xend(),
```

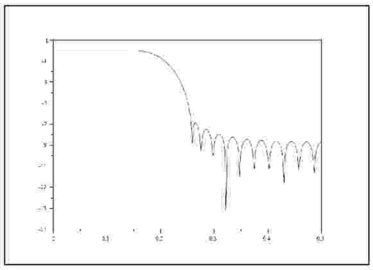

Figure 3.7: exec('fir7.code') Low pass filter with Kaiser window, $n = 33, \beta = 5.6$

```
//_____________________________________________
//FIR8.SCE
xinit('fir8.ps')
[wft,wfm,fr] = wfir('sb',127,[.2 .3],'hm',[0 0]);
plot(fr,log(wfm))
xend(),
```

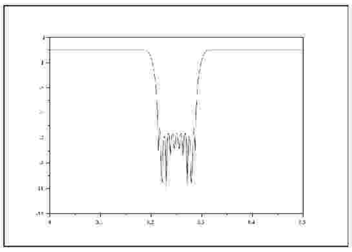

Figure 3.8: exec('fir8.code') Stop band filter with Hamming window, $n = 127, \alpha = .54$

```
//_____________________________________________
//FIR9.SCE
xinit('fir9.ps')
[wft,wfm,fr] = wfir('bp',55,[.15 .35],'ch',[.001 -1]);
plot(fr,log(wfm))
xend();
```

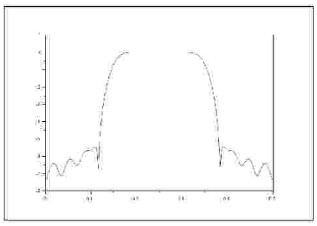

Figure 3.9: exec('fir9.code') Band pass filter with Chebyshev window, $n = 55$, $dp = .001$, $df = .0446622$

```
//_____________________________________________
//FSTYP121.SCE
xinit('fstyp121.ps')
hd = [0*ones(1,15) ones(1,10) 0*ones(1,39)];//desired samples
hst1 = fsfirlin(hd,1);//filter with no sample in the transition
hd(15) = .5;hd(26) = .5;//samples in the transition bands
hst2 = fsfirlin(hd,1);//corresponding filter
pas = 1/prod(size(hst1))*.5;
fg = 0:pas:.5;//normalized frequencies grid
n = prod(size(hst1))
plot(fg(1:n),hst1);
plot2d(fg(1:n)',hst2',[-3],"000");
```

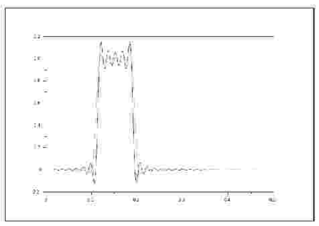

Figure 3.10: exec('fstyp121.code') Type 1 band pass filter with no sample or one sample in each transition band

```
//_____________________________________________
//FSTYP122.SCE
xinit('fstyp122.ps')
hd = ones(1,32);hd(65) = 0;//definition of samples
hst1 = fsfirlin(hd,1);//type 1 filter
hst2 = fsfirlin(hd,2);//type 2 filter
pas = 1/prod(size(hst1))*.5;
fg = pas:pas:.5;//normalized frequencies grid
plot2d([fg;fg]',[hst1;hst2]');
xend()
```

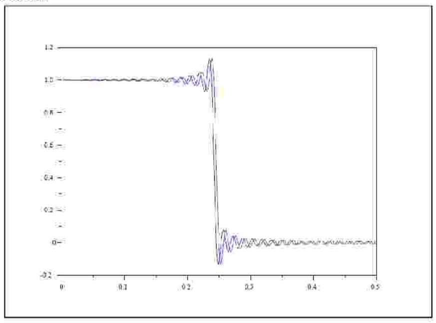

Figure 3.11: exec('fstyp122.code') Type 1 and type 2 low pass filter

```
//_____________________________________________
//REMEZ1.SCE
xinit('remez1.ps')
plot([0 12 0 0],[0 0 0 7]);
plot2d([0 4 9]',[4 0 6]',[-1],"000"),
plot2d([0 7.5 9]',[3 0 1]',[-1],"000"),
plot2d([0 2 9]',[2 0 7]',[-1],"000"),
plot2d([0 3 9]',[1 0 2.3333]',[-1],"000"),
x = (15/9)*(0:21)/21;
y = 4*ones(x)-x;
for k = 1:22,plot2d([x(k) x(k)+.07]',[y(k) y(k)+.07]',[-1],"000"),end
x = (15/9)*ones(1:19)+(75/21-15/9)*(0:18)/18;
y = 3*ones(x)-(6/15)*x;
for k = 1:19,plot2d([x(k) x(k)+.037]',[y(k) y(k)+.093]',[-1]',"000"),end
x = (75/21)*ones(1:59)+(8-75/21)*(0:58)/58;
y = -2*ones(x)+x;
for k = 1:59,plot2d([x(k) x(k)-.07]',[y(k) y(k)+.07]',[-1],"000"),end
t = 2*%pi*(0:35)/35;
st = .05*sin(t);
ct = .05*cos(t);
//plot(75/21*ones(t)+st,(75/21-2)*ones(t)+ct)
//plot2d([75/21 75/21]',[0 75/21-2]',[-1],"000"),
xstring(9,6.8,'|x-2|')
xstring(9,6,'|x-4|')
```

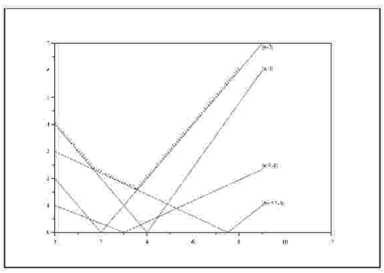

Figure 3.12: exec('remez1.code') Minimax Approximation for Linear Equations

```
  xstring(9,2.4,'|x/3-1|')
  xstring(9,1,'|6x/15-3|')
  xend(),
//________________________________________________________________
//REMEZ2_4.SCE
  xinit('remez2.ps');
//set up data for fortran subroutine remez
  nc = 21;
  ngrid = nc*250;
//frequency grid with no unspecified band
  fg = .5*(0:(ngrid-1))/(ngrid-1);
//desired function (low-pass filter)
  ds(1:ngrid/2) = ones(1:ngrid/2);
  ds(ngrid/2+1:ngrid) = 0*ones(1:ngrid/2);
//weight function
  wt = ones(fg);
//call remez
  an = remezb(nc,fg,ds,wt);
//obtain other half of filter coefficients (by symmetry)
  h = an(nc:-1:2)/2;
  h(nc) = an(1);
  h(nc+1:2*nc-1) = h(nc-1:-1:1);
//plot output
  z = poly(0,'z');
  hz = poly(h,'z','c');
  fr = (0:.5:.5*155)/156;
  rep = abs(freq(hz,1,exp(%i*2*%pi*fr)));
  hm = rep';
  plot(fr,hm),
  xend(),
  xinit('remez3.ps');
  clear fg ds wt nc an
  nc = 21;
  ngrid = nc*16;
//frequency grid with unspecified band
  fg = (0:-1+ngrid/2)*.24*2/(ngrid-2);
  fg(ngrid/2+1:ngrid) = fg(1:ngrid/2)+.26*ones(1:ngrid/2);
//desired function (low-pass filter)
  ds(1:ngrid/2) = ones(1:ngrid/2);
  ds(ngrid/2+1:ngrid) = 0*ones(1:ngrid/2);
//weight function
  wt = ones(fg);
//call remez
  an = remezb(nc,fg,ds,wt);
//obtain other half of filter coefficients (by symmetry)
  h = an(nc:-1:2)/2;
  h(nc) = an(1);
  h(nc+1:2*nc-1) = h(nc-1:-1:1);
//plot output
  hz = poly(h,'z','c');
  fr = (0:.5:.5*155)/156;
  rep = abs(freq(hz,1,exp(%i*2*%pi*fr)));
  hm = rep';
  plot(fr,hm),
  xend(),
  xinit('remez4.ps');
  clear nc ngrid fg ds wt;
  nc = 21;
  ngrid = nc*16;
//frequency grid with no unspecified band
  fg = .5*(0:(ngrid-1))/(ngrid-1);
//desired function (triangular)
  ds(1:ngrid/2) = (0:-1+ngrid/2)*2/(ngrid-2);
  ds(ngrid/2+1:ngrid) = ds(ngrid/2:-1:1);
//weight function
  wt = ones(fg);
```

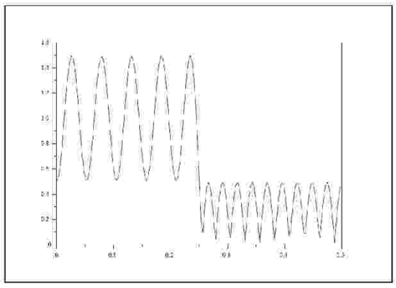

Figure 3.13: exec('remez2_4.code') Low Pass Filter with No Transition Band

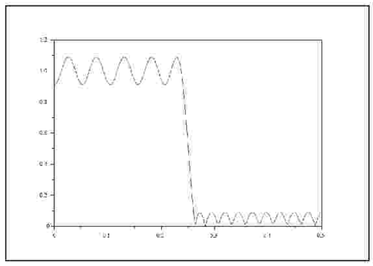

Figure 3.14: exec('remez2_4.code') Low Pass Filter with Transition Band [.24 .26]

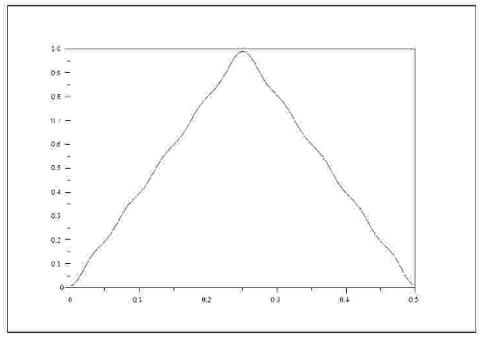

Figure 3.15: exec('remez2_4.code') Triangular Shaped Filter

```
//call remez
  an = remezb(nc,fg,ds,wt);
//obtain other half of filter coefficients (by symmetry)
  h = an(nc:-1:2)/2;
  h(nc) = an(1);
  h(nc+1:2*nc-1) = h(nc-1:-1:1);
//plot output
  hz = poly(h,'z','c');
  fr = (0:.5:.5*155)/156;
  rep = abs(freq(hz,1,exp(%i*2*%pi*fr)));
  hm = rep';
  plot(fr,hm),
  xend(),
//___________________________________________________________________
//REMEZ5_7.SCE
  xinit('remez5.ps');
  deff('[ ] = pf(h,nc)',...
  'hz = poly(h,"z","c");...
  fr = (0:.5:.5*155)/156;...
  rep = abs(freq(hz,1,exp(%i*2*%pi*fr)));...
  plot(fr,rep),')
  nf = 32;
  bedge = [0 .2;.22 .28;.3 .5];
  des = [1 0 1];
  wate = [1 1 1];
  hn = eqfir(nf,bedge,des,wate);
  pf(hn,nf);
  xend(),
  xinit('remez6.ps');
  nf = 33;
  bedge = [0 .2;.22 .28;.3 .5];
  des = [1 0 1];
  wate = [1 1 1];
  hn = eqfir(nf,bedge,des,wate);
  pf(hn,nf);
  xend(),
  xinit('remez7.ps');
  nf = 33;
  bedge = [0 .25;.28 .5];
  des = [0 1];
  wate = [1 1];
  hn = eqfir(nf,bedge,des,wate);
  pf(hn,nf);
  xend(),
//___________________________________________________________________
//ANALOG1.SCE
  xinit('analog1.ps')
  exec('Sanalog1.sce')
  xend()
//SANALOG1.SCE
//squared magnitude response of Butterworth filter
  h = buttmag(13,300,1:1000);
  mag = 20*log(h)'/log(10);
  plot2d((1:1000)',mag,[-1],"011"," ",[0,-180,1000,20]),
//___________________________________________________________________
//ANALOG2.SCE
  xinit('analog2.ps')
  exec('Sanalog2.sce')
  xend()
//SANALOG2.SCE
//Butterworth filter; 13 poles
  n = 13;
  angles = ones(1,n)*(%pi/2+%pi/(2*n))+(0:n-1)*%pi/n;
  s = exp(%i*angles); //Location of the poles
  xset("mark",0,1);
  lim = 1.2*sqrt(2.);
```

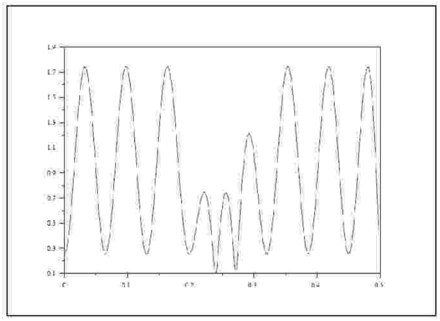

Figure 3.16: exec('remez5_7.code') Stop Band Filter of Even Length

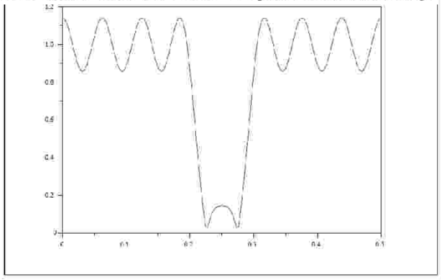

Figure 3.17: exec('remez5_7.code') Stop Band Filter of Odd Length

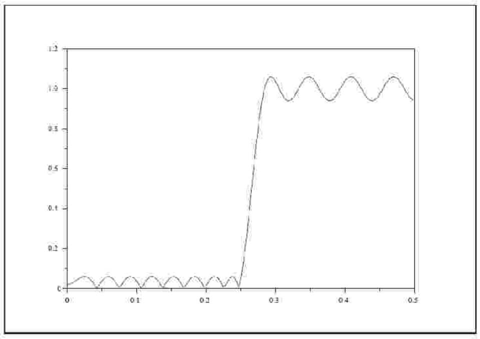

Figure 3.18: exec('remez5_7.code') High Pass Filter Design

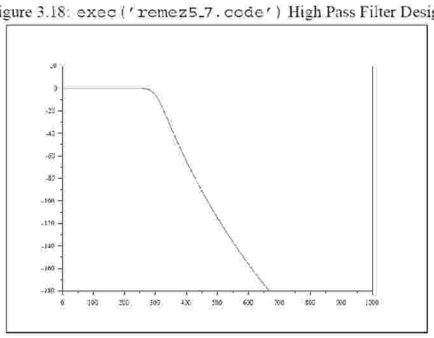

Figure 4.1: exec('analog1.code') Magnitude in dB. $n = 13, \omega_c = 300$

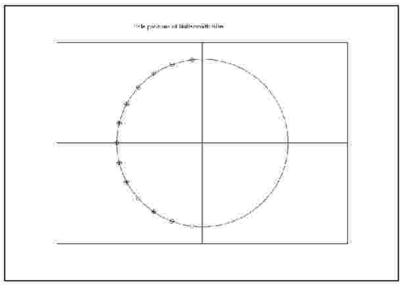

Figure 4.2: exec('analog2.code') Butterworth filter: pole positions. $n = 13$

```
plot2d(real(s)',imag(s)',[3],"012"," ",[-lim,-1.2,lim,1.2]);
xarc(-1,1,2,2,0,360*64);
xsegs([-lim,0;lim,0],[0,-1.2;0,1.2])
xtitle('Pole positions of Butterworth filter');
//
//ANALOG3.SCE
xinit('analog3.ps')
exec('Sanalog3.sce')
xend()
//SANALOG3.SCE
//Chebyshev; ripple in the passband
n = 13;epsilon = 0.2;omegac = 3;sample = 0:0.05:10;
h = cheb1mag(n,omegac,epsilon,sample);
plot(sample,h,'frequencies','magnitude')
//
```

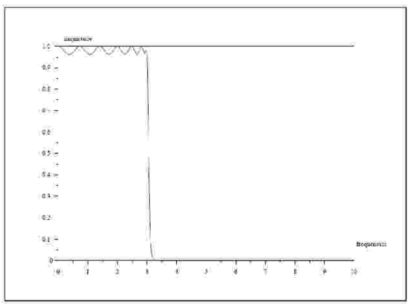

Figure 4.3: exec('analog3.code') Magnitude of a Type 1 Chebyshev filter

```
ANALOG4.SCE
xinit('analog4.ps')
n = 13;epsilon = 0.2;omegac = 3;sample = 0:0.05:10;
[p,gain] = zpch1(n,epsilon,omegac);
//Transfer function computation tr_fct(s) = gain/deno(s)
tr_fct = poly(gain,'s','coef')/real(poly(p,'s'))
//Magnitude of the frequency response computed along the
//imaginary axis for the values %i*sample...
rep = abs(freq(tr_fct(2),tr_fct(3),%i*sample));
plot(sample,rep,'frequencies','magnitude')
xend()
//
```

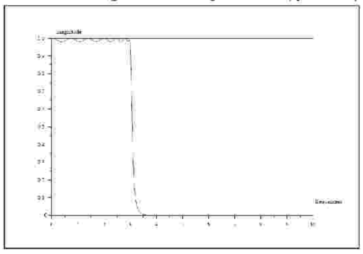

Figure 4.4: exec('analog4.code') Chebyshev filter: frequency response in magnitude

```
//ANALOG5.SCE
xinit('analog5.ps')
exec('Sanalog5.sce')
xend()
SANALOG5.SCE
//Chebyshev; ripple in the stopband
n = 10;omegar = 6;A = 1/0.2;sample = 0.0001:0.05:10;
h2 = cheb2mag(n,omegar,A,sample);
plot(sample,log(h2)/log(10),'frequencies','magnitude in dB')
//Plotting of frequency edges
minval = (-maxi(-log(h2)))/log(10);
plot2d([omegar;omegar],[minval;0],[-1],"000");
//Computation of the attenuation in dB at the stopband edge
attenuation = -log(A*A)/log(10);
plot2d(sample',attenuation*ones(sample)',[-2],"000")
//
```

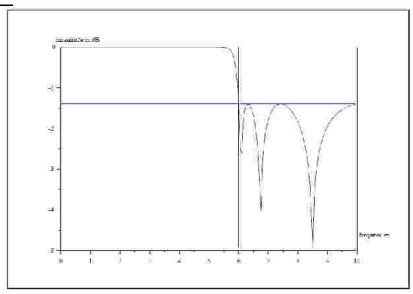

Figure 4.5: exec('analog5.code') Magnitude of a Type 2 Chebyshev filter

```
//ANALOG6.SCE
xinit('analog6.ps')
exec('Sanalog6.sce')
xend()
//SANALOG6.SCE
//The rectangle R0
m = 0.8+%eps;
z = %asn(1/sqrt(m),m);
K = real(z);KT = imag(z);
x2max = 1/sqrt(m);
x1 = 0:0.05:1;x2 = 1:((x2max-1)/20):x2max;x3 = x2max:0.05:10;
x = [x1,x2,x3];
rect = [0,-KT,1.1*K,2*KT]
y = %asn(x,m);
plot2d(real(y)',imag(y)',[-1],"011"," ",rect);
xtitle(' ','real(y)','imag(y)')
[n1,n2] = size(x)
x1 = 0:0.5:1;x2 = 1:0.3:x2max;x3 = x2max:1:10;
x1 = [0,0.25,0.5,0.75,1.0,1.1,1.2,1.3,1.4,2,3,4,10]
rect = [0,-KT,1.1*K,2*KT]
y1 = %asn(x1,m);
xnumb(real(y1),imag(y1)+0.1*ones(imag(y1)),x1)
plot2d(real(y1)',imag(y1)',[2],"011"," ",rect);
```

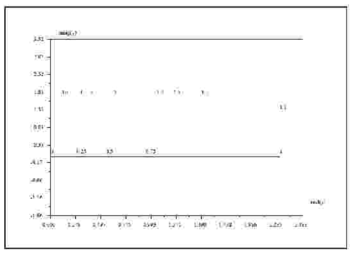

Figure 4.6 exec('analog6.code') The rectangle R_0, image by u of the positive real axis

```
//
//ANALOG7.SCE
 xinit('analog7.ps')
 exec('Sanalog7.sce')
 xend()
//SANALOG7.SCE
 m = 0.36; //m = k^2
 K = %k(m);
 P = 4*K; //Real period
 real_val = 0:(P/50):P;
 plot(real_val,real(%sn(real_val,m)),'x real','sn(x)')
```

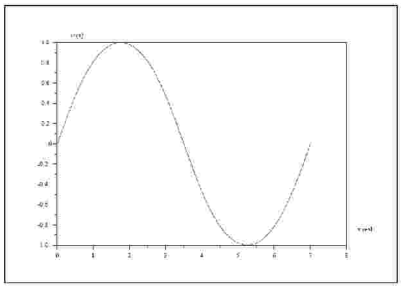

Figure 4.7: exec('analog7.code') Behavior of the sn function for real values

```
//
//ANALOG8.SCE
 xinit('analog8.ps')
 exec('Sanalog8.sce')
 xend()
//SANALOG8.SCE
 m = 0.36; //m = k^2
 KT = %k(1-m);
 Ip = 2*KT; //Imaginary period
 ima_val1 = [0.:(Ip/50):(KT-0.01)];
 ima_val2 = [(KT+0.01):(Ip/50):(Ip+KT)];
 z1 = %sn(%i*ima_val1,m);z2 = %sn(%i*ima_val2,m);
 rect = [0,-30,Ip+KT,30];
 plot2d([KT,KT]',[-30,30]',[-1],"011",' ',rect);
 xtitle(' ','x imaginary','sn(x)') //asymptote
 plot2d([-30,30]',[0,0]',[-1],"000");
 plot2d(ima_val1',imag(z1)',[-1],"000");
 plot2d(ima_val2',imag(z2)',[-1],"000");
```

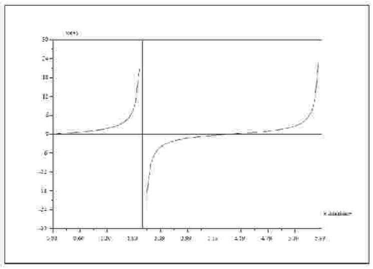

Figure 4.8: exec('analog8.code') Behavior of the sn function for imaginary values

```
//
//ANALOG9.SCE
 xinit('analog9.ps')
 n = 9;eps = 0.2;A = 3;m1 = eps*eps/(A*A-1);
 K1 = %k(m1);K1T = %k(1-m1);
 z1max = n*K1;z2max = K1T;
 z1 = 0:(z1max/100):z1max;
 z2 = %i*(0:(z2max/50):z2max);z2 = z2+z1max*ones(z2);
 z3 = z1max:-(z1max/100):0;z3 = z3+%i*z2max*ones(z3);
 plot(ell1mag(eps,m1,[z1,z2,z3]));
 omc = prod(size(z1));
 omr = prod(size([z1,z2]));
 plot2d([omc,omc]',[0,1]',[-2],"000");
 plot2d([omr,omr]',[0,1]',[-2],"000");
```

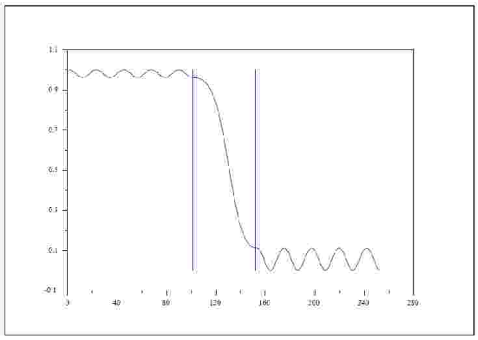

Figure 4.9: exec('analog9.code') v(z) for z in Σ_n, with $n = 9$

```
//
//ANALOG10.SCE
 xinit('analog10.ps')
 mm1 = 0:0.01:1;mm1(1) = 0.00000001;mm1(101) = 0.9999;
 m = 0*mm1;n = 3;i = 1;
 anorm = 1.-2.*%eps;
 for m1 = mm1,
   y = %asn(anorm/sqrt(m1),m1);
   K1 = real(y);
   K12 = imag(y);
   chi1 = K12/K1;
   m(i) = findm(chi1/n);
   i = i+1;
 end,
 plot(real(log(mm1)),real(log(m))),
```

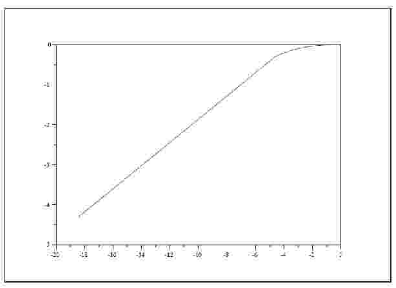

Figure 4.10: exec('analog10.code') log(m) versus log(m_1) for order n fixed

```
//
//ANALOG11.SCE
 xinit('analog11.ps')
 deff('[alpha,beta] = alpha_beta(n,m,m1)',...
 'if 2*int(n/2) = n then, beta = K1; else, beta = 0;end;...
 alpha = %k(1-m1)/%k(1-m);')
 epsilon = 0.1;
 A = 10; //ripple parameters
```

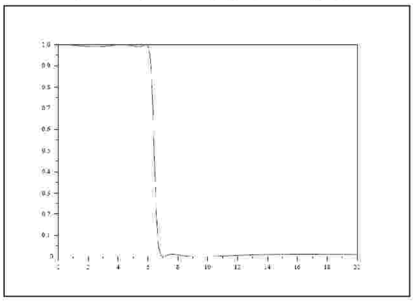

Figure 4.11: exec('analog11.code') Response of Prototype Elliptic Filter

```
 m1 = (epsilon*epsilon)/(A*A-1);n = 5;omegac = 6;
 m = find_freq(epsilon,A,n);
 omegar  =  omegac/sqrt(m)
 %k(1-m1)*%k(m)/(%k(m1)*%k(1-m))-n //Check...
 [alpha,beta] = alpha_beta(n,m,m1)
 alpha*%asn(1,m)-n*%k(m1) //Check
 sample = 0:0.01:20;
//Now we map the positive real axis into the contour...
 z = alpha*%asn(sample/omegac,m)+beta*ones(sample);
 plot(sample,ell1mag(epsilon,m1,z))
//
//ANALOG12.SCE
 xinit('analog12.ps')
 exec('Sanalog12.sce')
 xend()
//SANALOG12.SCE
//Filter with zpell
 epsilon = 0.1;A = 10; //ripple parameters
 m1 = (epsilon*epsilon)/(A*A-1);n = 5;omegac = 6;
 m = find_freq(epsilon,A,n);
 omegar  =  omegac/sqrt(m)
 [z,p,g] = zpell(epsilon,A,omegac,omegar);
//Now computes transfer function
 num = real(poly(z,'s'));den = real(poly(p,'s'));
 transfer = g*num/den
//Plot of the response
 sample = 0:0.01:20;
 rep = freq(g*num,den,%i*sample);
 plot(sample,abs(rep))
//
```

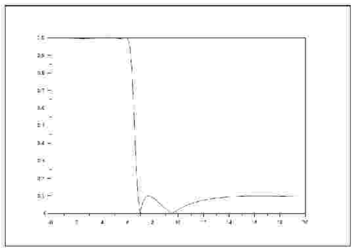

Figure 4.12: exec('analog12.code') Example of response of a filter obtained by zpell

```
//IIR1.SCE
 xinit('iir1.ps')
//make figure 1 for iir.tex
 deff('[ ] = hatch_c(c1,c2,r,nl)',...
 'a = -sqrt(2)*r:2*sqrt(2)*r/(nl+1):sqrt(2)*r;...
 r = r*ones(a);...
 c1 = c1*ones(a);...
 c2 = c2*ones(a);...
 xh = c1+(a+sqrt(2*r.*r-a.*a))/2;...
 xl = c1+(a-sqrt(2*r.*r-a.*a))/2;...
 yh = c2-c1+xh-a;...
 yl = c2-c1+xl-a;...endfunction
 for k = 1:maxi(size(a)),...
    plot2d(real([xl(k) xh(k)])'',real([sqrt(2)*yl(k) sqrt(2)*yh(k)])'',[-1],'''000'''),...
 end,')
 deff('[ ] = hatch_r(x1,y1,x2,y2,nl)',...
 'a = x1-y2+y1:(x2-x1-y1+y2)/(nl+1):x2;...
 for k = 1:maxi(size(a)),...
   if a(k)<x1 then,...
     xl(k) = x1;...
     yl(k) = y1+x1-a(k);...
   else,...
     xl(k) = a(k);...
     yl(k) = y1;...
   end,...
   if a(k)<x2-y2+y1 then,...
     xh(k) = a(k)+y2-y1;...
     yh(k) = y2;...
   else,...
     xh(k) = x2;...
     yh(k) = y1+xh(k)-a(k);...
   end,...
 end,...
 for k = 1:maxi(size(a)),...
   plot2d(real([xl(k) xh(k)])'',real([yl(k) yh(k)])'',[-1],'''000'''),...
 end,')
```

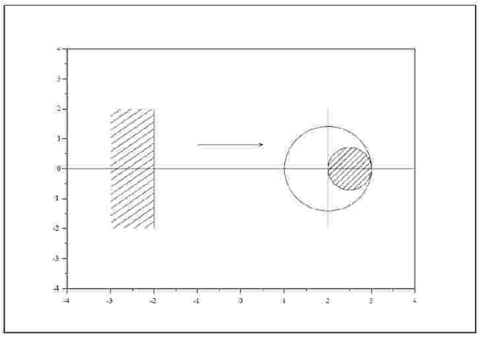

Figure 4.13: exec('iir1.code') Transform $s = (1 - z^{-1})/T$

```
 t = 0:.1:.1+2*%pi;
 c = cos(t);
 s = sin(t);
 plot([-4 -4 4],[4 -4 -4]),
 plot2d([-4 4 -2 -2 -2 -2 2 2 2]',[0 0 0 2 -2 0 0 2 -2]',[-1],"000"),
 plot2d((c+2*ones(t))',sqrt(2)*s',[-1],"000"),
 plot2d((.5*c+2.5*ones(t))',.5*sqrt(2)*s',[-1],"000"),
 hatch_c(2.5,0,.5,10);
 hatch_r(-3,-2,-2,2,20);
 plot2d([-1 .5]',[.8 .8]',[-1],"000"),
 plot2d([.4 .5]',[.85 .8]',[-1],"000"),
 plot2d([.4 .5]',[.75 .8]',[-1],"000"),
 xend(),
//
 IIR2_3.SCE
 xinit('iir2.ps')
 dess(31) = 1;
 [pols,gn] = zpch1(3,.22942,2);
 hs = gn/real(poly(pols,'s'));
 fr = 0:.05:3*%pi;
 hsm = abs(freq(hs(2),hs(3),%i*fr));
 plot(fr,hsm)
 xend(),
 xinit('iir3.ps')
 z = poly(0,'z');
 hz = horner(hs,2*(z-1)/(z+1));
 fr = 0:.005:.5;
 hzm = abs(freq(hz(2),hz(3),exp(2*%pi*%i*fr)));
 plot(fr,hzm)
 xend(),
//
//IIR4.SCE
 xinit('iir4.ps')
 hz = iir(5,'lp','cheb1',[.2 0],[.05 .05]);
 fr = 0:.002:.5;
 hzm = abs(freq(hz(2),hz(3),exp(2*%pi*%i*fr)));
 plot(fr,hzm)
 xend()
//
//IIR5.SCE
 xinit('iir5.ps')
 hz = iir(3,'bp','ellip',[.15 .25],[.08 .03]);
 fr = 0:.002:.5;
 hzm = abs(freq(hz(2),hz(3),exp(2*%pi*%i*fr)));
 plot(fr,hzm)
 xend()
//
//EQIIR1.SCE
 xinit('eqiir1.ps')
 om = [0.251463,1*%pi/10,2*%pi/10,0.773302];
 deltap = 0.022763;
 deltas = 0.01;
 [cells,fact,zers,pols] = eqiir('bp','el',om,deltap,deltas);
 n = prod(cells(2));d = prod(cells(3));
 rep = freq(n,d,exp(%i*(0:0.01:%pi)));
 rep = fact*abs(rep);
 n = prod(size(rep))
 plot(20*log(rep(2:n))/log(10))
 xend(),
//
//EQIIR4.SCE
 xinit('eqiir4.ps')
 exec('Seqiir4.sce')
 xend(),
//SEQIIR4.SCE
//Elliptic bandpass filter
```

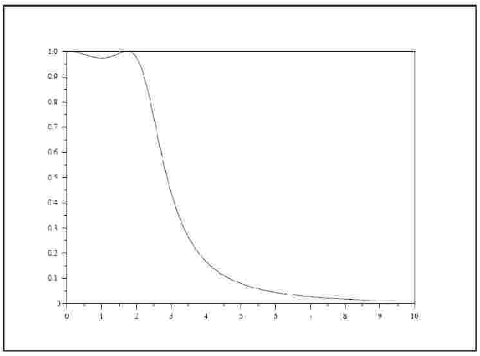

Figure 4.14: exec('iir2_3.code') Magnitude of Analog Filter

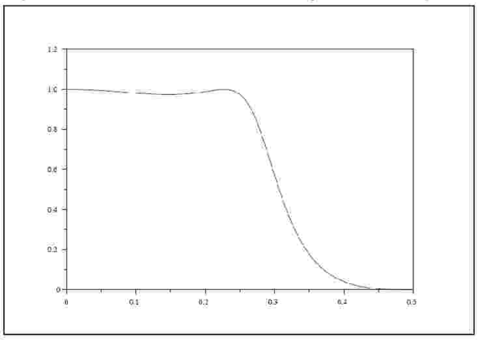

Figure 4.15: exec('iir2_3.code') Magnitude of Digital Filter

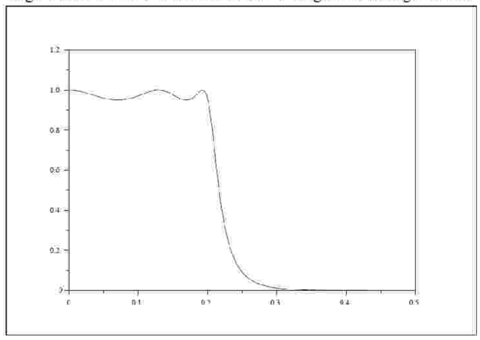

Figure 4.16: exec('iir4.code') Digital Low-Pass Filter

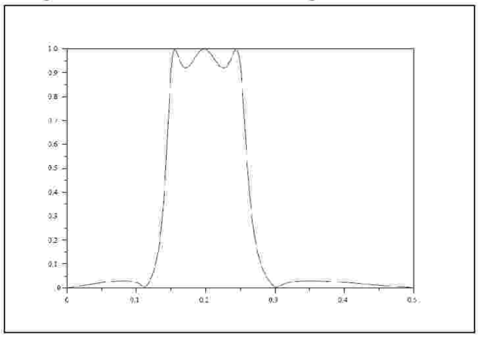

Figure 4.17: exec('iir5.code') Digital Band-Pass Filter

```
 om = [0.251463,1*%pi/10,2*%pi/10,0.773302];
 deltap = 0.022763;
 deltas = 0.01;
 [cells,fact,zers,pols] = eqiir('bp','el',om,deltap,deltas);
 n = prod(cells(2));d = prod(cells(3));
 rep = freq(n,d,exp(%i*(0:0.01:%pi)));
 rep = fact*abs(rep);
 n = prod(size(rep))
 plot(20*log(rep(2:n))/log(10))
//
```

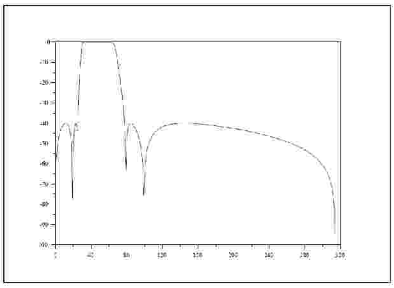

Figure 4.18 exec('eqiir4.code') Example of response obtained with eqiir

```
//SPECT1.SCE
 xinit('spect1.ps')
//get some random looking data
 rand('normal');
 rand('seed',0);
 x = rand(1:256-33+1);
// [h,w] = ffir('pb',33,.1,'tr');
 [h,w] = wfir('lp',33,[.1,0],'tr',[0.01,-1]);
 h1 = [h 0*ones(1:maxi(size(x))-1)];
 x1 = [x 0*ones(1:maxi(size(h))-1)];
 hf = fft(h1,-1);
 xf = fft(x1,-1);
 yf = hf.*xf;
 y = real(fft(yf,1));
//plot frame
 rect = [1,-2.35,256,1.4];
 n = prod(size(y));
 plot2d((1:n)',y',[-1],"011",' ',rect)
//vertical bars
 plot2d([100 100]',[-1.5 -1.7]',[-1],"000")
 plot2d([78 78]',[-1.8 -2]',[-1],"000")
 plot2d([178 178]',[-1.8 -2]',[-1],"000")
 plot2d([156 156]',[-2.1 -2.3]',[-1],"000")
//horizontal bars
 plot2d([1 35]',[-1.6 -1.6]',[-1],"000")
 plot2d([65 100]',[-1.6 -1.6]',[-1],"000")
 plot2d([78 113]',[-1.9 -1.9]',[-1],"000")
 plot2d([143 178]',[-1.9 -1.9]',[-1],"000")
 plot2d([156 191]',[-2.2 -2.2]',[-1],"000")
 plot2d([221 256]',[-2.2 -2.2]',[-1],"000")
//put in x1, x2, and x3
 xstring(42,-1.65,'x1')
 xstring(120,-1.95,'x2')
 xstring(198,-2.25,'x3')
//draw arrow heads
 plot2d([1 5]',[-1.6 -1.57]',[-1],"000")
 plot2d([1 5]',[-1.6 -1.63]',[-1],"000")
 plot2d([78 82]',[-1.9 -1.87]',[-1],"000")
 plot2d([78 82]',[-1.9 -1.93]',[-1],"000")
 plot2d([156 160]',[-2.2 -2.17]',[-1],"000")
 plot2d([156 160]',[-2.2 -2.23]',[-1],"000")
 plot2d([96 100]',[-1.57 -1.6]',[-1],"000")
 plot2d([96 100]',[-1.63 -1.6]',[-1],"000")
 plot2d([174 178]',[-1.87 -1.9]',[-1],"000")
 plot2d([174 178]',[-1.93 -1.9]',[-1],"000")
 plot2d([252 256]',[-2.17 -2.2]',[-1],"000")
 plot2d([252 256]',[-2.23 -2.2]',[-1],"000")
 xend(),
//
```

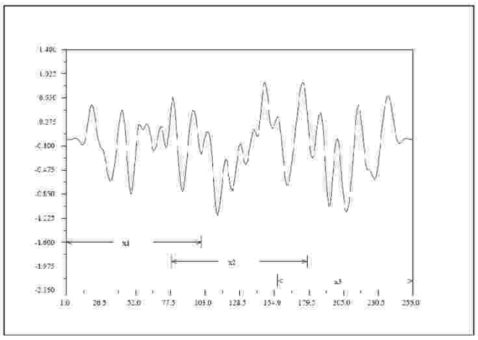

Figure 5.1: exec('spect1.code') Overlapping Data

```
//SPECT2_4.SCE
 xinit('spect2.ps')
//test modified periodogram method
//and correlation method spectral estimation techniques
 dess(31) = 1;
//generate white data
 rand('normal');
```

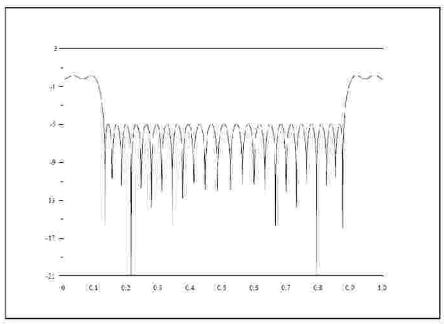

Figure 5.2: exec('spect2_4.code') Log Magnitude Squared of Filter

```
 rand('seed',0);
 x = rand(1:1024-33+1);
//make low-pass filter with eqfir
 nf = 33;
 bedge = [0 .1;.125 .5];
 des = [1 0];
 wate = [1 1];
 h = eqfir(nf,bedge,des,wate);
//filter white data to obtain colored data
 h1 = [h 0*ones(1:maxi(size(x))-1)];
 x1 = [x 0*ones(1:maxi(size(h))-1)];
 hf = fft(h1,-1);
 xf = fft(x1,-1);
 yf = hf.*xf;
 y = real(fft(yf,1));
//plot magnitude of filter
 h2 = [h 0*ones(1:968)];
 hf2 = fft(h2,-1);
 hf2 = real(hf2.*conj(hf2));
 hsize = maxi(size(hf2));
 fr = (1:hsize)/hsize;
 plot(fr,log(hf2));
 xend()
 xinit('spect3.ps')
//pspect example
 [sm] = pspect(100,200,'tr',y);
 smsize = maxi(size(sm));
 fr = (1:smsize)/smsize;
 plot(fr,log(sm))
 xend(),
 xinit('spect4.ps')
//cspect example
 [sm] = cspect(100,200,'tr',y);
 smsize = maxi(size(sm));
 fr = (1:smsize)/smsize;
 plot(fr,log(sm))
 xend()
//________________________________________________________________________
//MEM1_3.SCE
 xinit('mem1.ps')
//define macro which computes the Blackman-Tukey periodogram
 deff('[xm,fr] = bt(x)',...
 'xsize = maxi(size(x));...
 [xf,fr] = frmag(x,256);...
 xm = xf.*conj(xf)/xsize;')
//rand('seed',12345),
 y = (0:10);
 x1 = sin(2*%pi*y/20);
 x2 = sin(3.5*%pi*y/20);
 w = .4*(rand(y)-.5*ones(y));
 x = x1+2*x2+w;
 [sm,fr1] = mese(x,10);
 [xm,fr] = bt(x);
 plot(x);
 xend()
 xinit('mem2.ps')
 plot(fr1,sm)
 xend()
 xinit('mem3.ps')
 plot(fr,xm)
 xend()
//________________________________________________________________________
//KF1.SCE
 xinit('kf1.ps')
 exec('Skf1.sce')
 xend(),
```

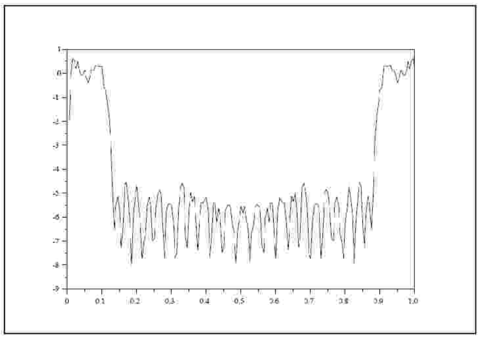

Figure 5.3: exec('spect2_4.code') Estimate of Spectrum

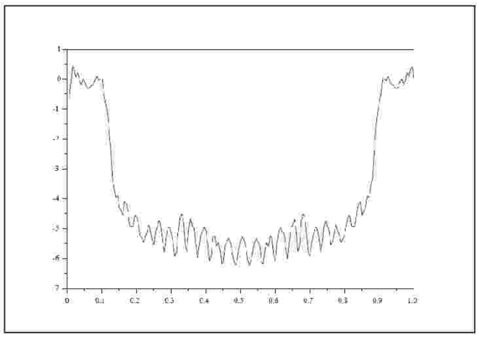

Figure 5.4: exec('spect2_4.code') Estimate of Spectrum

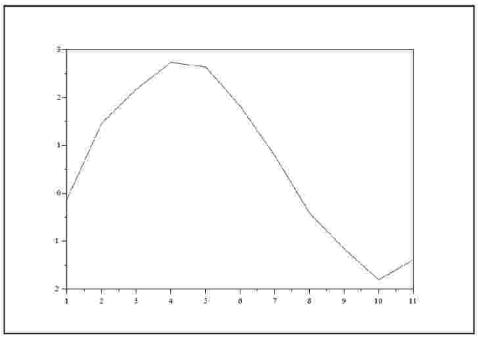

Figure 5.5: exec('mem1_3.code') Input Data Sequence, $x(n)$

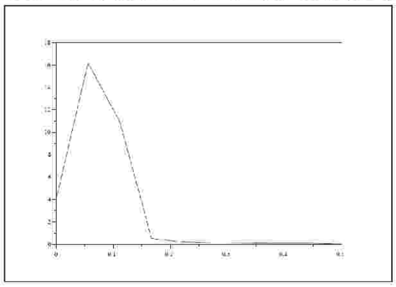

Figure 5.6: exec('mem1_3.code') Maximum Entropy Spectral Estimate of $x(n)$

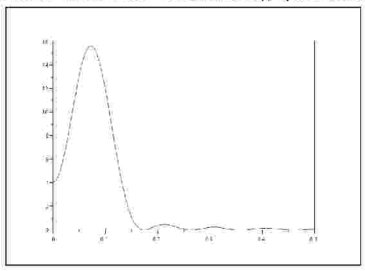

Figure 5.7: exec('mem1_3.code') Squared Magnitude of the Fourier Transform of $x(n)$

```
//SKF1.SCE
//test of the steady-state kalman filter
 rand('seed',5);rand('normal');
 q = [.03 .01;.01 .03];u = rand(2,11);
 f = [1.1 .1;0 .8];g = (chol(q))';
 m0 = [10 10]';p0 = [2 0;0 2];x0 = m0+(chol(p0))'*rand(2,1);
 x = ltitr(f,g,u,x0);
 r = [2 0;0 2];v = (chol(r))'*rand(2,11);y = x+v;
 h = eye(2,2);[xe] = sskf(y,f,h,q,r,m0);
//plot result
 a = mini([x(1,:),xe(1,:)]);a = -.1*abs(a)+a;
 b = maxi([x(1,:),xe(1,:)]);b = .1*abs(b)+b;
 c = mini([x(2,:),xe(2,:)]);c = -.1*abs(c)+c;
 d = maxi([x(2,:),xe(2,:)]);d = .1*abs(d)+d;
//plot frame, real state (x), and estimate (xke)
 plot([a a b],[d c c]),
 plot2d(x(1,:)',x(2,:)',[-1],'000',' ')
 plot2d(xe(1,:)',xe(2,:)',[-2],'000',' '),
 plot2d(xe(1,:)',xe(2,:)',[3],'000',' '),
//
```

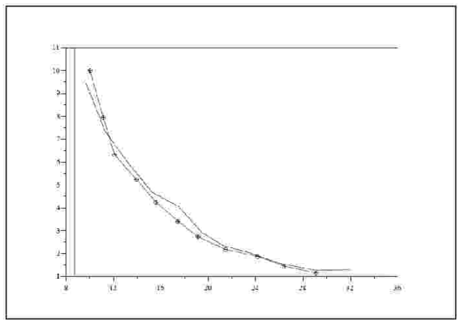

Figure 6.1: exec('kf1.code') Steady-State Kalman Filter Tracking

```
//KF2.SCE
 xinit('kf2.ps')
 exec('Skf2.sce')
 xend(),
//SKF2.SCE
//generate test process to be sent to kalman filter
//initialize state statistics (mean and err. variance)
 m0 = [10 10]';p0 = [2 0;0 2];
//create system
 f = [1.1 .1;0 .8];g = [1 0;0 1];h = [1 0;0 1];
//noise statistics
 q = [.03 .01;.01 .03];r = 2*eye(2,2);
//initialize system process
 rand('seed',2);rand('normal');
 p0c = chol(p0);x0 = m0+p0c'*rand(ones(m0));yt = [ ];
//initialize kalman filter
 xke0 = m0;pk0 = p0;
//initialize plotted variables
 x = x0;xke = m0;
 ell = [pk0(1,1) pk0(2,2) pk0(1,2)]';
//loop
 n = 10;
 for k = 1:n,
//generate the state and observation at time k (i.e. x(k+1) and y(k))
 [x1,y] = system(x0,f,g,h,q,r);
 x = [x x1];
 yt = [yt y];
 x0 = x1;
//track the state with the standard kalman filter
 [xke1,pk1,xd,pd] = kalm(y,xke0,pk0,f,g,h,q,r);
 xke = [xke xke1];
 ell = [ell [pk1(1,1) pk1(2,2) pk1(1,2)]'];
 xke0 = xke1;
 pk0 = pk1;
//end loop
 end,
//define macro which traces an ellipse
 deff('[ ] = ellipse(m1,m2,s1,s2,s12)',...
 't = 0:.1:.1+%pi*2;...
 c = 2*cos(t);...
 s = 2*sin(t);...
 rho = s12/sqrt(s1*s2);...
 cr = sqrt(s1)*c+m1*ones(c);...
 sr = sqrt(s2)*(rho*c+sqrt(1-rho*rho)*s)+m2*ones(s);...
 plot2d(cr'',sr'',[-1],''000''),')
//plot result
```

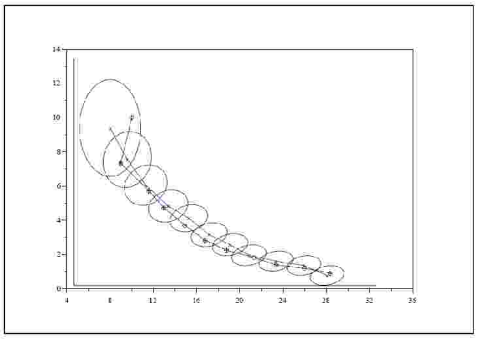

Figure 6.2: exec('kf2.code') Kalman Filter Tracking

```
  a = mini([x(1,:)-2*sqrt(ell(1,:)),xke(1,:)]);a = -.1*abs(a)+a;
  b = maxi([x(1,:)+2*sqrt(ell(1,:)),xke(1,:)]);b = .1*abs(b)+b;
  c = mini([x(2,:)-2*sqrt(ell(2,:)),xke(2,:)]);c = -.1*abs(c)+c;
  d = maxi([x(2,:)+2*sqrt(ell(2,:)),xke(2,:)]);d = .1*abs(d)+d;
//plot frame, real state (x), and estimate (xke)
  plot([a a b],[d c c]),
  plot2d(x(1,:)',x(2,:)',[-2],"000"),
  plot2d(xke(1,:)',xke(2,:)',[-1],"000"),
//plot ellipses of constant likelihood (2 standard dev's)
  for k = 1:n+1,
  ellipse(x(1,k),x(2,k),ell(1,k),ell(2,k),ell(3,k)),
  end,
//mark data points (* for real data, o for estimates)
  plot2d(x(1,:)',x(2,:)',[2],"000"),
  plot2d(xke(1,:)',xke(2,:)',[3],"000")
//_______________________________________________________________
//WF1.SCE
  xinit('wf1.ps')
  exec('Swf1.sce')
  xend(),
//SWF1.SCE
//test of the wiener filter function
// initialize state statistics (mean and er. variance)
  m0 = [10 10]';p0 = [100 0;0 100];
// create system
  f = [1.15 .1;0 .8];g = [1 0;0 1];
  h = [1 0;0 1];[hi,hj] = size(h);
// noise statistics
  q = [.01 0;0 .01];r = 20*eye(2,2);
// initialize system process
  rand('seed',66);rand('normal');
  p0c = chol(p0);x0 = m0+p0c'*rand(ones(m0));
  y = h*x0+chol(r)'*rand(ones(1:hi))';yt = y;
// initialize plotted variables
  x = x0;
// loop
  ft = [f];gt = [g];ht = [h];qt = [q];rt = [r];
  n = 10;
  for k = 1:n,
// generate the state and observation at time k (i.e. xk and yk)
  [x1,y] = system(x0,f,g,h,q,r);
  x = [x x1];yt = [yt y];x0 = x1;
  ft = [ft f];gt = [gt g];ht = [ht h];
  qt = [qt q];rt = [rt r];
// end loop
  end;
// get the wiener filter estimate
  [xs,ps,xf,pf] = wiener(yt,m0,p0,ft,gt,ht,qt,rt);
// plot result
  a = mini([x(1,:)-2*sqrt(ps(1,1:2:2*(n+1))),xf(1,:),xs(1,:)]);
  b = maxi([x(1,:)+2*sqrt(ps(1,1:2:2*(n+1))),xf(1,:),xs(1,:)]);
  c = mini([x(2,:)-2*sqrt(ps(2,2:2:2*(n+1))),xf(2,:),xs(2,:)]);
  d = maxi([x(2,:)+2*sqrt(ps(2,2:2:2*(n+1))),xf(2,:),xs(2,:)]);
  xmargin = maxi([abs(a),abs(b)]);
  ymargin = maxi([abs(c),abs(d)]);
  a = -.1*xmargin+a;b = .1*xmargin+b;
  c = -.1*ymargin+c;d = .1*ymargin+d;
// plot frame, real state (x), and estimates (xf, and xs)
  plot([a a b],[d c c]);
  plot2d(x(1,:)',x(2,:)',[-2],"000"),
  plot2d(xf(1,:)',xf(2,:)',[-2],"000"),
  plot2d(xs(1,:)',xs(2,:)',[-2],"000"),
// mark data points (* for real data, o for estimates)
  plot2d(xs(1,:)',xs(2,:)',[2],"000"),
  plot2d(xf(1,:)',xf(2,:)',[3],"000"),
  plot2d(x(1,:)',x(2,:)',[4],"000"),
```

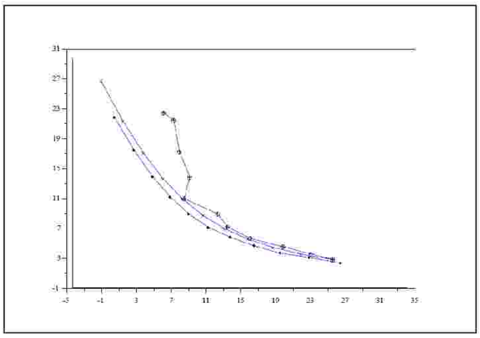

Figure 6.3: exec('wf1.code') Wiener Smoothing Filter

```
//______________________________________________________________
//OPTIIR1.SCE
 xinit('optiir1.ps')
 [ce0,f0,ze0,po0] = eqiir('lp','ellip',%pi*[.5;.65;0;0],.1,.01);
 hz0 = f0*prod(ce0(2))./prod(ce0(3));
 ze0 = ze0(1:2:4);po0 = po0(1:2:4);
 x0 = [abs([ze0 po0])';atan(imag([ze0 po0]),real([ze0 po0]))';10];
 x = x0;
 omega = %pi/100:%pi/100:%pi;
 p = 1;
 wa(1:52) = ones(1,52);
 wa(53:100) = .5*ones([53:100]);
 rp0 = abs(freq(hz0(2),hz0(3),exp(%i*omega)));
 cx = 'normalized frequency';
 cy = 'magnitude';
 plot(rp0)
 xstring(0.1,1.01,'magnitude');
 xstring(70,-.12,'normalized frequency');
 xend()
```

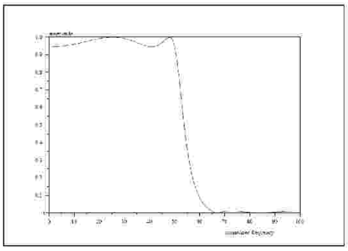

Figure 7.1: exec('optiir.1.code') Minimum mean-square design. Fourth order IIR filter

```
//______________________________________________________________
//OPTIIR2.SCE
 xinit('optiir2.ps')
 [ce0,f0,ze0,po0] = eqiir('lp','ellip',%pi*[.5;.65;0;0],.1,.01);
 hz0 = f0*prod(ce0(2))./prod(ce0(3));
 ze0 = ze0(1:2:4);po0 = po0(1:2:4);
 x0 = [abs([ze0 po0])';atan(imag([ze0 po0]),real([ze0 po0]))';10];
 x = x0;
 omega = %pi/100:%pi/100:%pi;
 p = 1;
 wa(1:52) = ones(1,52);
 wa(53:100) = .5*ones([53:100]);
 rp0 = abs(freq(hz0(2),hz0(3),exp(%i*omega)));
 r = 1;
 ld = 20*log(r.*rp0)/log(10);
 cx = 'normalized frequency';
 cy = 'magnitude in dB';
 plot(ld);
 xstring(2,1.01,'magnitude in dB');
 xstring(72,-90,'normalized frequency');
 xend()
```

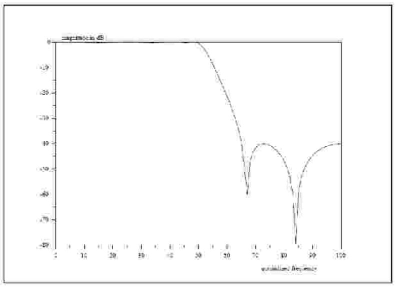

Figure 7.2: exec('optiir.2.code') Resulting magnitude response. Log scale

```
//______________________________________________________________
//OPTIIR3.SCE
//design of a low-pass filter with normalized discrete frequency .25
// ripple in the passband 0.1, ripple in the stopband 0.01,
// transition bandwidth .1
 [ce0,f0,ze0,po0] = eqiir('lp','ellip',%pi*[.5;.65;0;0],.1,.01);
 hz0 = f0*prod(ce0(2))./prod(ce0(3));
 ze0 = ze0(1:2:4);po0 = po0(1:2:4);
 x0 = [abs([ze0 po0])';atan(imag([ze0 po0]),real([ze0 po0]))';10];
 x = x0;
 omega = %pi/100:%pi/100:%pi;
 p = 1;
 wa(1:52) = ones(1,52);
 wa(53:100) = .5*ones([53:100]);
 rp0 = abs(freq(hz0(2),hz0(3),exp(%i*omega)));
//plot(rp0);
//xbasc();
 ad(1:49) = ones(1,49)./rp0(1:49);
 ad(50:100) = rp0(50:100);
 x = [x0(1:4) x0(5:8)];
 [cout,xx1,grad,to] = optim(iirmod,x);
 if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
 if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
 if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
 if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
 [cout,xx1,grad,to] = optim(iirmod,xx1);
```

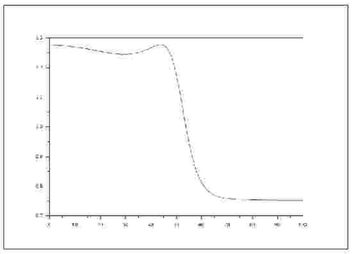

Figure 7.3: exec('optiir.3.code') Minimum mean-square design. Sixth order IIR filter

```
if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
binf = [0;-2*%pi].*.ones(4,1);
bsup = [1;2*%pi].*.ones(4,1);
binf = [binf(1:4) binf(5:8)]
bsup = [bsup(1:4) bsup(5:8)]
[cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,x);
[cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,xx2);
z = poly(0,'z');
z1 = xx2(1,1)*exp(%i*xx2(1,2));
z2 = xx2(2,1)*exp(%i*xx2(2,2));
num = (z-z1)*(z-z1')*(z-z2)*(z-z2')
num = real(num);
p1 = xx2(3,1)*exp(%i*xx2(3,2));
p2 = xx2(4,1)*exp(%i*xx2(4,2));
den = (z-p1)*(z-p1')*(z-p2)*(z-p2');
den = real(den);
sl = syslin('c',num/den);
ff = repfreq(sl,0.01,0.5,0.01);
rp1 = abs(freq(num,den,exp(%i*omega)));
xinit('optiir3.ps')
plot(rp1);
xend();
//plot(rp0);
//xbasc();
//xinit('optiir4.ps')
//plot(20.*log(rp0.*rp1));
//xend();
//____________________________________________________________
//OPTIIR4.SCE
//design of a low-pass filter with normalized discrete frequency .25
// ripple in the passband 0.1, ripple in the stopband 0.01,
// transition bandwidth .1
[ce0,f0,ze0,po0] = eqiir('lp','ellip',%pi*[.5;.65;0;0],.1,.01);
hz0 = f0*prod(ce0(2))./prod(ce0(3));
ze0 = ze0(1:2:4);po0 = po0(1:2:4);
x0 = [abs([ze0 po0])';atan(imag([ze0 po0]),real([ze0 po0]))';10];
x = x0;
omega = %pi/100:%pi/100:%pi;
p = 1;
wa(1:52) = ones(1,52);
wa(53:100) = .5*ones([53:100]);
rp0 = abs(freq(hz0(2),hz0(3),exp(%i*omega)));
//plot(rp0);
//xbasc();
ad(1:49) = ones(1,49)./rp0(1:49);
ad(50:100) = rp0(50:100);
x = [x0(1:4) x0(5:8)];
[cout,xx1,grad,to] = optim(iirmod,x);
if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
[cout,xx1,grad,to] = optim(iirmod,xx1);
if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
binf = [0;-2*%pi].*.ones(4,1);
bsup = [1;2*%pi].*.ones(4,1);
binf = [binf(1:4) binf(5:8)]
bsup = [bsup(1:4) bsup(5:8)]
[cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,x);
[cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,xx2);
```

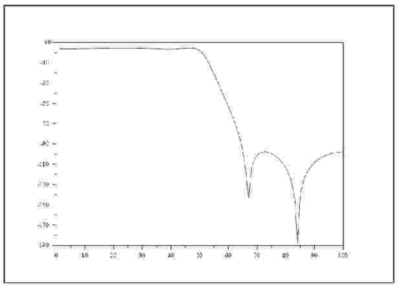

Figure 7.4: exec('optiir.4.code') Resulting magnitude response. Log scale

```
 z = poly(0,'z');
 z1 = xx2(1,1)*exp(%i*xx2(1,2));
 z2 = xx2(2,1)*exp(%i*xx2(2,2));
 num = (z-z1)*(z-z1')*(z-z2)*(z-z2')
 num = real(num);
 p1 = xx2(3,1)*exp(%i*xx2(3,2));
 p2 = xx2(4,1)*exp(%i*xx2(4,2));
 den = (z-p1)*(z-p1')*(z-p2)*(z-p2');
 den = real(den);
 sl = syslin('c',num/den);
 ff = repfreq(sl,0.01,0.5,0.01);
 rp1 = abs(freq(num,den,exp(%i*omega)));
//plot(rp1);
//plot(rp0);
//xbasc();
 xinit('optiir4.ps')
 plot(20.*log(rp0.*rp1));
 xend()
//____________________________________________________________________________
```

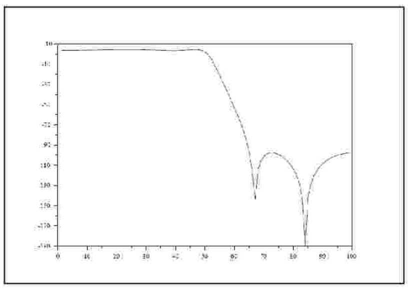

Figure 7.5: exec('optiir.5.code') Log-magnitude response. $\omega \in [0, 0.45]$

```
//OPTIIR5.SCE
//design of a low-pass filter with normalized discrete frequency .25
// ripple in the passband 0.1, ripple in the stopband 0.01,
// transition bandwidth .1
 [ce0,f0,ze0,po0] = eqiir('lp','ellip',%pi*[.5;.65;0;0],.1,.01);
 hz0 = f0*prod(ce0(2))./prod(ce0(3));
 ze0 = ze0(1:2:4);po0 = po0(1:2:4);
 x0 = [abs([ze0 po0])';atan(imag([ze0 po0]),real([ze0 po0]))';10];
 x = x0;
 omega = %pi/100:%pi/100:%pi;
 p = 1;
 wa(1:52) = ones(1,52);
 wa(53:100) = .5*ones([53:100]);
 rp0 = abs(freq(hz0(2),hz0(3),exp(%i*omega)));
//plot(rp0);
//xbasc();
 ad(1:49) = ones(1,49)./rp0(1:49);
 ad(50:100) = rp0(50:100);
 x = [x0(1:4) x0(5:8)];
 [cout,xx1,grad,to] = optim(iirmod,x);
 if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
 if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
 if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
 if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
 [cout,xx1,grad,to] = optim(iirmod,xx1);
 if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
 if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
 if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
 if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
 binf = [0;-2*%pi].*.ones(4,1);
 bsup = [1;2*%pi].*.ones(4,1);
 binf = [binf(1:4) binf(5:8)]
 bsup = [bsup(1:4) bsup(5:8)]
 [cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,x);
 [cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,xx2);
 z = poly(0,'z');
 z1 = xx2(1,1)*exp(%i*xx2(1,2));
 z2 = xx2(2,1)*exp(%i*xx2(2,2));
 num = (z-z1)*(z-z1')*(z-z2)*(z-z2')
 num = real(num);
 p1 = xx2(3,1)*exp(%i*xx2(3,2));
 p2 = xx2(4,1)*exp(%i*xx2(4,2));
 den = (z-p1)*(z-p1')*(z-p2)*(z-p2');
 den = real(den);
 sl = syslin('c',num/den);
 ff = repfreq(sl,0.01,0.225,0.01);
 rp1 = abs(freq(num,den,exp(%i*omega)));
```

```
//plot(rp1);
//plot(rp0);
//xbasc();
 xinit('optiir5.ps')
 plot(20.*log(rp0.*rp1));
 xend()
//______________________________________________________________________
//OPTIIRTOTAL.SCE
//design of a low-pass filter with normalized discrete frequency .25
// ripple in the passband 0.1, ripple in the stopband 0.01,
// transition bandwidth .1
 [ce0,f0,ze0,po0] = eqiir('lp','ellip',%pi*[.5;.65;0;0],.1,.01);
 hz0 = f0*prod(ce0(2))./prod(ce0(3));
 ze0 = ze0(1:2:4);po0 = po0(1:2:4);
 x0 = [abs([ze0 po0])';atan(imag([ze0 po0]),real([ze0 po0]))';10];
 x = x0;
 omega = %pi/100:%pi/100:%pi;
 p = 1;
 wa(1:52) = ones(1,52);
 wa(53:100) = .5*ones([53:100]);
 rp0 = abs(freq(hz0(2),hz0(3),exp(%i*omega)));
 plot(rp0);
 xbasc();
 ad(1:49) = ones(1,49)./rp0(1:49);
 ad(50:100) = rp0(50:100);
 x = [x0(1:4) x0(5:8)];
 [cout,xx1,grad,to] = optim(iirmod,x);
 if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
 if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
 if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
 if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
 [cout,xx1,grad,to] = optim(iirmod,xx1);
 if xx1(1,1) > 1. then xx1(1,1) = 1/xx1(1.1); end;
 if xx1(2,1) > 1. then xx1(2,1) = 1/xx1(2,1); end;
 if xx1(3,1) > 1. then xx1(3,1) = 1/xx1(3,1); end;
 if xx1(4,1) > 1. then xx1(4,1) = 1/xx1(4,1); end;
 binf = [0;-2*%pi].*.ones(4,1);
 bsup = [1;2*%pi].*.ones(4,1);
 binf = [binf(1:4) binf(5:8)]
 bsup = [bsup(1:4) bsup(5:8)]
 [cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,x);
 [cout,xx2,grad,to] = optim(iirmod,'b',binf,bsup,xx2);
 z = poly(0,'z');
 z1 = xx2(1,1)*exp(%i*xx2(1,2));
 z2 = xx2(2,1)*exp(%i*xx2(2,2));
 num = (z-z1)*(z-z1')*(z-z2)*(z-z2')
 num = real(num);
 p1 = xx2(3,1)*exp(%i*xx2(3,2));
 p2 = xx2(4,1)*exp(%i*xx2(4,2));
 den = (z-p1)*(z-p1')*(z-p2)*(z-p2');
 den = real(den);
 sl = syslin('c',num/den);
 ff = repfreq(sl,0.01,0.5,0.01);
 rp1 = abs(freq(num,den,exp(%i*omega)));
 plot(rp1);
 plot(rp0);
 xbasc();
 plot(20.*log(rp0.*rp1));
//______________________________________________________________________
//OPTFIR1.SCE
 xinit('optfir1.ps');
//An example of design of a low-pass filter using frequency-sampling
//design and linear programming optimization
//cut-off frequency = .3 and 3 samples in the transition band
 forder = 64; // filter length
 Nf = 64; // filter length
```

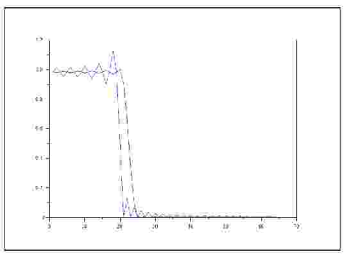

Figure 7.6 exec('optfir1.code') Linear programming design. 64-point lowpass FIR filter

```
  Nf2 = Nf/2;tol = 0.05;N = 128;N2 = N/2;
//simple rectangular filter
  Hsimp = 0*ones(N,1);
  for i = 1:20
    Hsimp(i) = 1.;
    Hsimp(N+1-i) = 1;
  end;
  FF = fft(Hsimp,1);
//plot(real(FF));
  FFF = 0*ones(N,1);
  for i = 1:Nf2
    FFF(i) = FF(i);
    FFF(N+1-i) = FF(N+1-i);
  end;
  FF1 = fft(FFF,-1);
  RR1 = real(FF1);
//plot(abs(RR1));
//Bounds for the design
  Hd = 0*ones(Nf,1);
  for i = 1:20
    Hd(i) = 1.;
  end;
  for i = 24:Nf
    Hd(i) = .006;
  end;
//Matrix of the interpolating functions
  S = 0*ones(N2,Nf);
  for ii = 1:N2
    for kk = 1:Nf
      i = ii-1;
      k = kk-1;
      omega = 2.*%pi*i/N;
      omega2 = %pi*i/N;
      coe = exp(-%i*(omega2-k*%pi/Nf))*(Nf-1)/2;
      ccc = (omega2-%pi*k/Nf);
      cc1 = sin(Nf*ccc);
      cc2 = sin(ccc);
      if cc2  =  0 then cc12 = Nf;else cc12 = cc1/cc2;end;
        S(ii,kk) = real(coe*cc12);
      end;
    end;
    for ii = 1:N2
      for kk = 1:Nf2
        Y(ii,kk) = S(ii,kk)+S(ii,Nf+1-kk);
      end;
    end;
    HU = 0*ones(Nf2,1);
    for i = 1:10
      HU(i) = 1.;
    end;
  B = Y*HU/Nf;
  B = B/Nf2;
//plot(B)
  A1 = Y(:,11);
  A2 = Y(:,12);
  A3 = Y(:,13);
  A = [A1 A2 A3];
  T = [0.75 ; 0.50 ; 0.25; 0.1];
  A4 = 0*A1;
  for i = 26:64
    A4(i) = -1.;
  end;
  bsup = Hd-B;
  binf = Hd-B;
  for i = 1:19
    bsup(i) = bsup(i)+tol;
```

```
   binf(i) = binf(i)-tol;
  end;
  for i = 1:6
   bsup(19+i) = 2.;
   binf(19+i) = -2.;
  end;
  for i = 26:64
   bsup(i) = 0.02;
   binf(i) = -0.02;
  end;
//Linear programming problem : CC*T < =  bound; min p'*T
//T = transition coefficients and cut-off bound
  p = [0;0;0;1];
  CC = [A A4; -A A4];
  CC = CC/32;
  CC = CC/128;
  bound = [bsup; -binf];
  ci = [0;0;0;0];
  cs = [1;1;1;0.05];
  [xx,yy,zz] = linpro(p,CC,bound,ci,cs);
  Hbest = HU;
  Hbest(11) = xx(1);
  Hbest(12) = xx(2);
  Hbest(13) = xx(3);
  Hres = Y*Hbest/Nf;
  Hres = Hres/32;
  xend();
//plot(abs(Hres));
  xinit('optfir1.ps');
  plot2d([1:64;1:64]',[abs(Hres) abs(B)])
  xend()
//______________________________________________________________
//OPTFIR2.SCE
  exec('optfir1.sce');
  xinit('optfir2.ps');
  plot2d([1:64;1:64]',[20*log(abs(Hres)) 20*log(abs(B))]);
//plot(20*log(abs(Hres)));
  xend();
//______________________________________________________________
//WIGNER1.SCE
  xinit('wigner1.ps')
// parabola
  a = [488**2 488 1;408**2 408 1;568**2 568 1];
  b = [1.28;0;0];
  x = a\b;
  t = 408:568;
  p = x'*[t.*t;t;ones(t)];
// unit step function
  u = [0*ones(408:487) ones(488:568)];
// finite duration sinusoid
  s = p.*sin(2*%pi/16*t+u*%pi);
// signal to be analyzed
  s = [0*ones(0:407) s 0*ones(569:951)];
// 64-point rectangular window
  h = ones(1,64);
// wigner spectrum
  w = wigner(s,h,12,128);
  plot3d(1:69,1:64,abs(w(1:69,1:64)));
  xend()
//______________________________________________________________
```

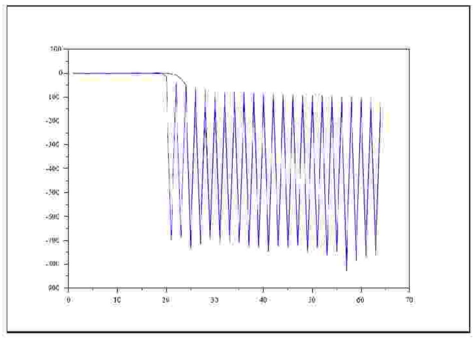

Figure 7.7: exec('optfir2.code') Linear programming design.

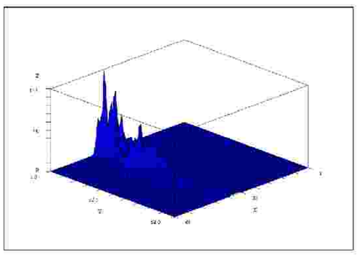

Figure 9.1: exec('wigner1.code') Wigner analysis. Sinusoid modulated by a parabola