Electronic and Communication Systems (Linear Circuits)

Assignment 1 - Computer system for filter design

Antonino Iannella, 9308390R
Supervisor: Anant Mahajan

Wednesday, 15 February 2023

All my work unless otherwise stated.

Ž®

CONTENTS

Index of figures and tables

Introduction

This aim of this assignment is to

- become familiar with various circuit analysis software, and
- develop a software tool to gain an understanding of various techniques and strategies used to develop such design packages.

A few circuit design packages are compared according to the features they offer. The software packages chosen to develop is a fourth-order Sallen-and-Key filter composed of two second-order filters. The system shall calculate the component values which the filter requires. The functions of the package are

- a choice of a lowpass or highpass filter response
- a choice of using the Butterworth or Chebyshev approximation
- the corner frequency, ω_o, selectable by the user
- the option to create the necessary Matlab code to plot the circuit's frequency response

Overview of existing software

Previous circuit design packages vary in features and quality. Analysis methods used differ between packages, some methods were found to be better than others. Most packages were developed to serve a particular industry field, like defence. Some packages are presented here with their creators.

PSPICE (MicroSim Corporation)

- dc, ac, and transient circuit analyses
- may perform any analysis using Laplace transform operators or Fast Fourier Transforms
- employs circuits models with parameters stored in libraries
- may calculate any symbolic functions even if it does not relate to electronic circuits, and may view any result graphically using PROBE

ASTAP (IBM Corporation)

- developed to handle the needs of Large Scale Integration (LSI) microchip technology
- dc, ac, and transient circuit analyses
- may use the dc analysis to find the operating point for ac analysis
- uses *Monte Carlo* analysis for input parameter variations like tolerances, and temperature

BELAC (General Electric Company)

- Uses network design techniques which is useful to engineers of many disciplines, not just Electronic Engineers
- FORTRAN routines may be used to model special components
- dc, ac, and transient circuit analyses for linear and non- linear networks
- the user only needs to describe the network to be calculated and the analysis required

CIRC (Xerox Corporation)

- consists of three packages; CIRC-DC (dc analysis), CIRC-AC (ac analysis), CIRC-TR (transient analysis)
- dc analysis features built-in non-linear Ebers-Moll parameters for transistors and diodes which require specification data as input, but yields accurate circuit designs
- may perform a worst-case analysis
- ac analysis handles passive and active components and has an accurate stability analysis
- transient analysis features good time dependence modelling

CIRCUS-2 (Boeing Company)

- finds dc steady-state and time-domain transient analysis with forcing function
- uses dc steady-state analysis as the initial conditions for time-domain analysis

ECAP II (IBM Corporation)

- dc, ac, and transient circuit analyses for linear and non- linear circuits using an advanced iteration technique
- single circuit description is used for any analysis
- input language statement may be modified at any time
- employs circuits models with parameters stored in libraries
- elements may be dependent on circuit properties like voltage and current
- circuit parameters may be user-defined
- diagnostic messages warn the user of any errors

LISA (IBM Corporation)

- analyses linear systems using Laplace (or s-plane) circuit methods

MARTHA (Massachusetts Institute of Technology)

performs frequency domain analysis of one or two-port networks
uses a generalised language for analysis

SCEPTRE (US Air Force Weapons Laboratory)

- programs may be easily written and modified
- uses many forms of input like constants, tables of data, or functions

- uses state-of-the-art numerical methods

SYSCAP (Rockwell International)

- may perform dc, ac, and transient circuit analyses as well as the radiation response, or finding the Fourier coefficients of the frequency response

Filter designer model

The system is an OS/2 Presentation Manager application. The GUI tool used is Dr Dialog from the IBM Corporation. The system uses the REXX programming language to perform all calculations.

The filter designer models a fourth-order filter. This filter is composed of two second-order Sallen-and-Key filters, each having unity amplifier gain. The coefficients of the terms of the transfer function is what is =needed by Matlab to plot the frequency response. The package allows the user to design a filter with

- a choice between a lowpass or highpass filter response,
- a choice of using the Butterworth or Chebyshev approximation,
- a selectable corner frequency, ω_o, and
- the option to create the necessary code to plot the circuit's frequency response using the Matlab® mathematical package.

The filter models used are shown below.

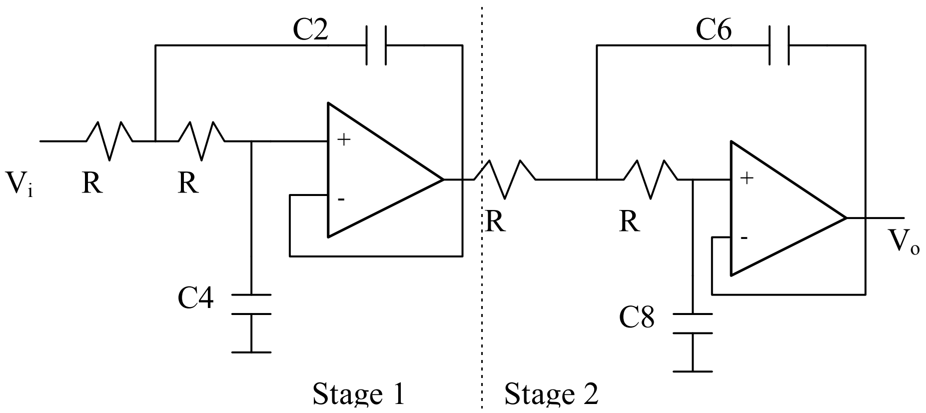

Figure 1 - Fourth-order lowpass filter with unity gain

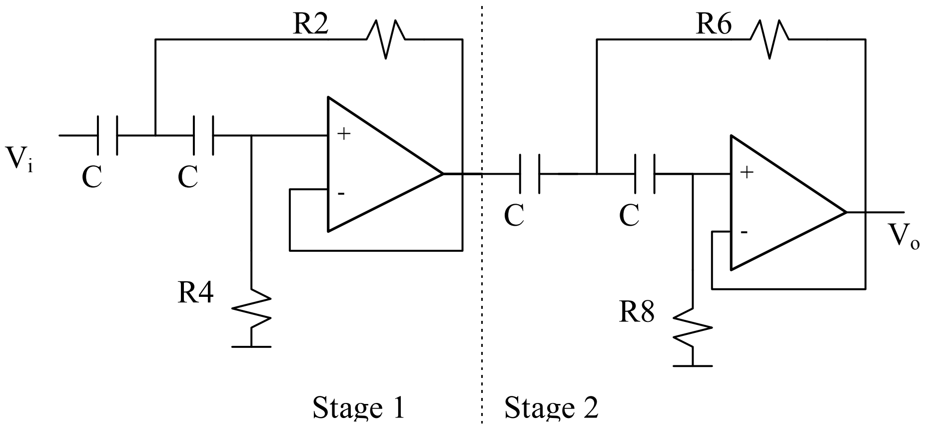

Figure 2 - Fourth-order highpass filter with unity gain

Design diagram

The flowchart representing the system is shown below. Due to the event-based nature of GUI applications, the flowchart's algorithm is not sequential.

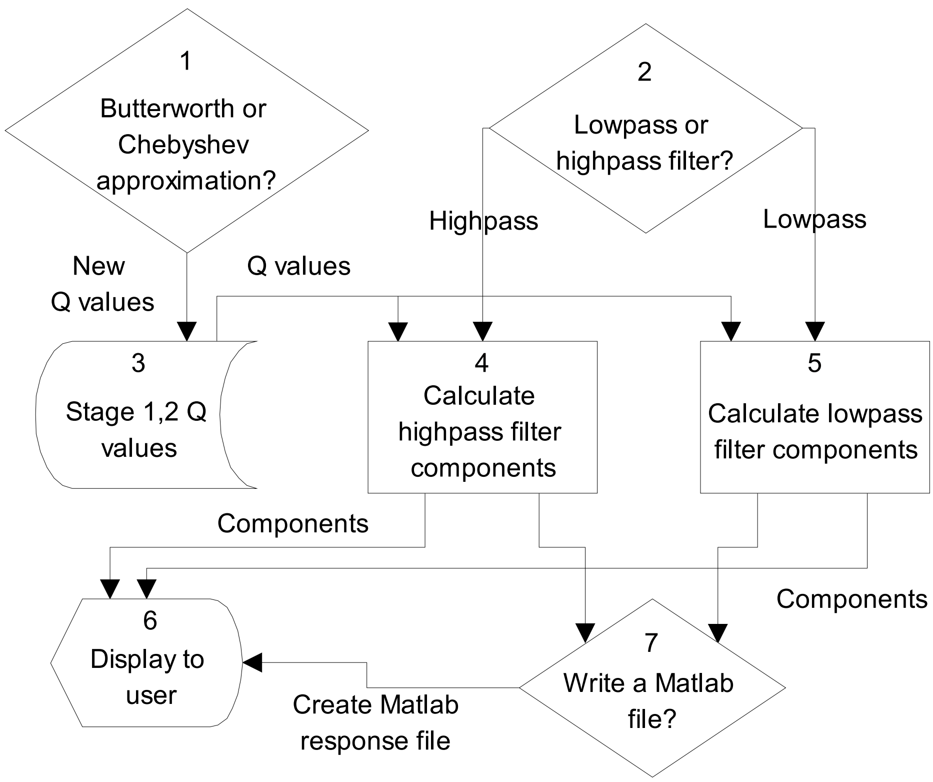

Figure 3 - System's functional diagram

System restrictions

The system's restrictions are

Functional

- the four resistors of the lowpass filter have an unchangeable value of R=10KΩ, and
- the four capacitors of the highpass filter have an unchangeable value of C=10nF. These values were selected as the filter defaults. The user is provided only with the necessary filter component values of
  - C2, C4, C6, C8 for the lowpass filter
  - R2, R4, R6, R8 for the highpass filter

  to complete the circuit.
- both filter stages have unity gain which is hard-coded into the system
- the quality factors of both the filter's second order stages are not selectable by the user, only by choosing the Butterworth or Chebyshev approximation

Non-functional

- the corner frequency is not allowed to be zero, and must be a positive integer. If it is negative, it is changed to a positive integer automatically. If it is zero or non-numeric, the user is warned of their mistake and the corner frequency reverts to its previous value.

The circuit components of the fourth-order filter may be easily calculated by using the properties of the contingent second-order stages. The capacitors of the lowpass filter (or resistors of the highpass filter) are found using the Quality Factors and the resistor values of their inherent stage, as well as the corner frequency value.

The corner frequency is *w*, R is the resistor value (10KΩ), and C is the capacitor value (10nF). Q1, and Q2 are the Quality Factors of stages 1 and 2 of the filter. The stages' quality factors are determined by choosing the Butterworth or Chebyshev approximation. The assumed values are

Table 1 - Q values for Butterworth and Chebyshev filter approximations

	Q1 (Stage 1)	*Q2 (Stage 2)*
Butterworth	0.54	1.31
Chebyshev	0.62	2.18

Lowpass filter formulae

The capacitors of the second-order lowpass filter stages are calculated using

$$C2 = 2\frac{Q1}{w\,R} \qquad C4 = \frac{1}{2}\frac{1}{Q1\,w\,R}$$

$$C6 = 2\frac{Q2}{w\,R} \qquad C8 = \frac{1}{2}\frac{1}{Q2\,w\,R}$$

The transfer function for stage 1 of the lowpass filter configuration is

$$TF = \frac{1}{R^2\,C2\,C4\left(s^2 + s\left(2\frac{1}{R\,C2}\right) + \frac{1}{R^2\,C2\,C4}\right)}$$

and similarly for stage 2

$$TF = \frac{1}{R^2\,C6\,C8\left(s^2 + s\left(2\frac{1}{R\,C6}\right) + \frac{1}{R^2\,C6\,C8}\right)}$$

Multiplying these obtains the fourth-order transfer function of the filter

$$\frac{Vo}{Vi} = \frac{\frac{1}{R^4C2C4C6C8}}{s^4 + s^3\left(\frac{2}{R}\left(\frac{1}{C2}+\frac{1}{C6}\right)\right) + s^2\left(\frac{1}{R^2}\left(\frac{1}{C6C8}+\frac{1}{C2C4}+\frac{4}{C2C6}\right)\right) + s\left(\frac{2}{R^3C2C6}\left(\frac{1}{C8}+\frac{1}{C4}\right)\right) + \frac{1}{R^4C2C4C6C8}}$$

Highpass filter formulae

The resistors of the second-order highpass filter stages are calculated using

$$R2 = \frac{1}{2}\frac{1}{Q1\,w\,C} \qquad R4 = 2\frac{Q1}{w\,C}$$

$$R6 = \frac{1}{2}\frac{1}{Q2\,w\,C} \qquad R8 = 2\frac{Q2}{w\,C}$$

The transfer function for stage 1 of the highpass filter configuration is

$$TF = \frac{s^2}{s^2 + s\left(2\frac{1}{R4\,C}\right) + \frac{1}{R2\,R4\,C^2}}$$

and similarly for stage 2

$$TF = \frac{s^2}{s^2 + s\left(2\,\frac{1}{R6\,C}\right) + \frac{1}{R6\,R8\,C^2}}$$

Multiplying these obtains the fourth-order transfer function of the filter

$$\frac{Vo}{Vi} = \frac{s^4}{s^4 + s^3\left(\frac{2}{C}\left(\frac{1}{R4}+\frac{1}{R8}\right)\right) + s^2\left(\frac{1}{C^2}\left(\frac{1}{R6R8}+\frac{1}{R2R4}+\frac{4}{R4R8}\right)\right) + s\left(\frac{2}{C^3R4R8}\left(\frac{1}{R6}+\frac{1}{R2}\right)\right) + \frac{1}{C^4R2R4R6R8}}$$

The user interface

An example of the application's user interface is shown with an example output.

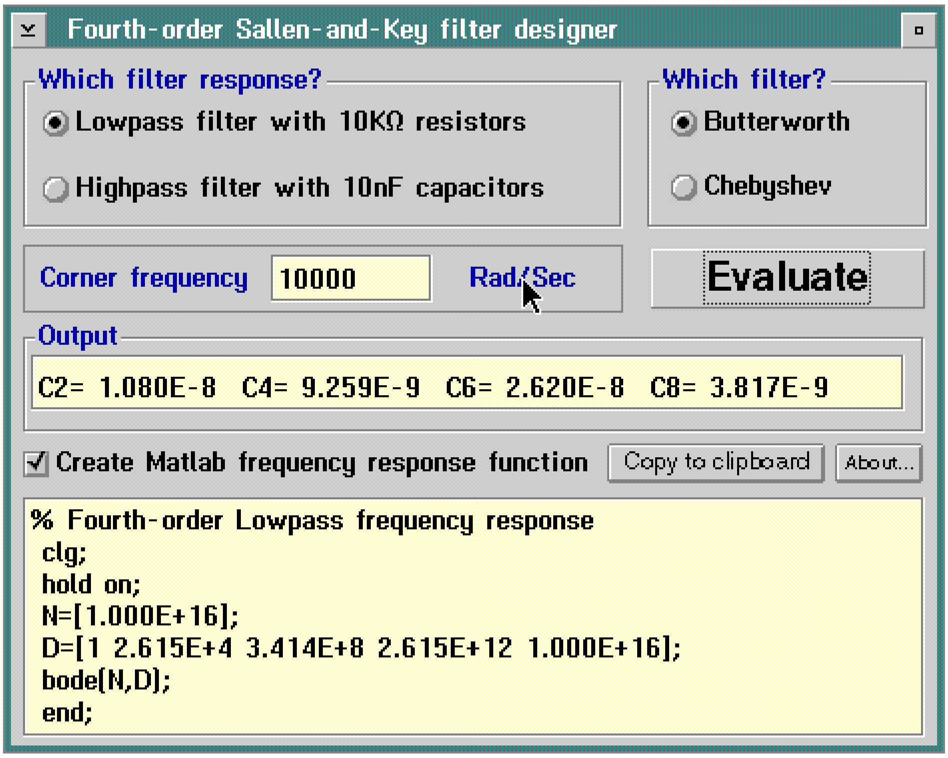

Figure 4 - User interface with an example output

Filter designer usage instructions

The user may obtain a set of component values by following these steps:

- Click on one of the radio buttons in the 'Which filter response?' box to select either the lowpass or highpass filter configuration.
- Click on one of the radio buttons in the 'Which filter?' box to select either the Butterworth or Chebyshev approximation.
- Type the desired corner frequency (in radians per second) in the text box on the right of the 'Corner frequency' label.
- Users who have Matlab may click on the 'Create Matlab frequency response function' (it is off by default) to write the necessary code to generate the Matlab bode plot.
- Press the 'Evaluate' button will calculate the selected component values and display them in the 'Output' box, and the Matlab function if the option is selected.
- Press the 'Copy to clipboard' button to copy the Matlab function to the clipboard. If the user has Matlab, they may view the output by pasting the program into the 'Matlab command window' and pressing the 'Enter' key.
- Pressing the 'About...' button at any time displays information about the author.

Performance evaluation

Four examples are provided. Based on the Matlab plots, the system yields the correct results;

- The lowpass and highpass responses are correct.
- The corner frequency for each plot is correct.
- The magnitude gain is 0 decibels, hence the overall gain is unity, as desired.
- The Butterworth approximations have a smooth response.
- The Chebyshev approximations have a ripple in the passband.

Lowpass filter, Butterworth approximation, ω=10000 rad/sec

Output:

```
C2= 1.080E-8  C4= 9.259E-9  C6= 2.620E-8  C8= 3.817E-9
```

Matlab function:

```
% Fourth-order Lowpass filter frequency response
clg;
hold on;
N=[1.000E+16];
D=[1 2.615E+4 3.414E+8 2.615E+12 1.000E+16];
bode(N,D);
end;
```

Matlab plot:

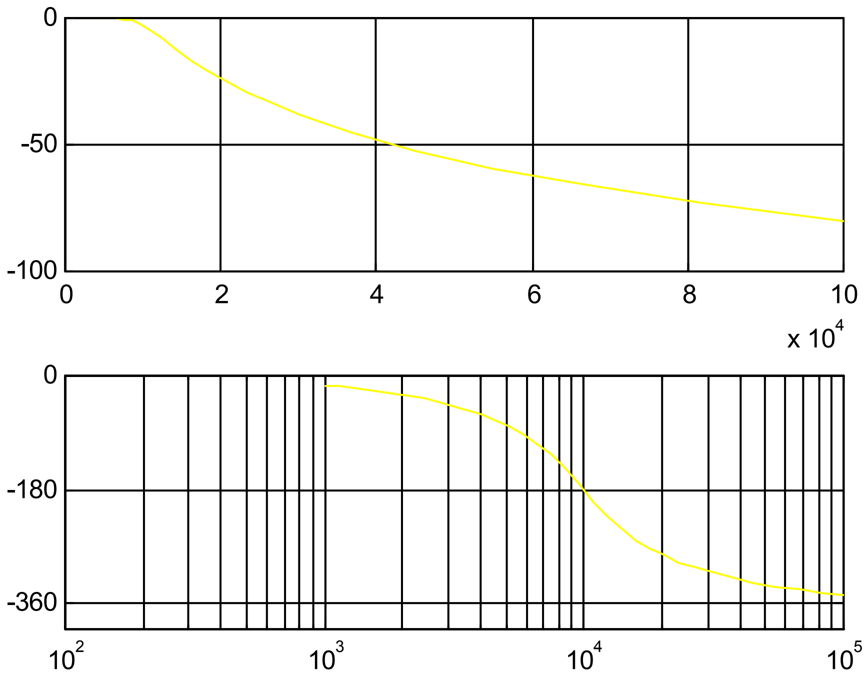

Figure 5 - Lowpass Butterworth response

Lowpass filter, Chebyshev approximation, ω=10000 rad/sec

Output:

```
C2= 1.240E-8  C4= 8.065E-9  C6= 4.360E-8  C8= 2.294E-9
```

Matlab function:

```
% Fourth-order Lowpass filter frequency response
clg;
hold on;
N=[1.000E+16];
D=[1 2.072E+4 2.740E+8 2.072E+12 1.000E+16];
bode(N,D);
end;
```

Matlab plot:

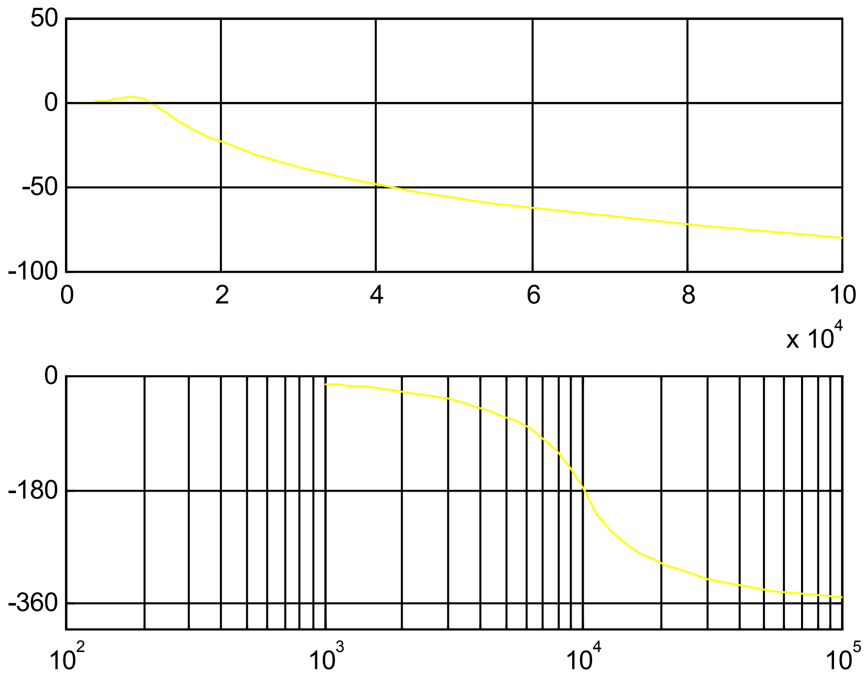

Figure 6 - Lowpass Chebyshev response

Highpass filter, Butterworth approximation, ω=5000 rad/sec

Output:

```
R2= 1.852E+4   R4= 2.160E+4   R6= 7.634E+3   R8= 5.240E+4
```

Matlab function:

```
% Fourth-order Highpass filter frequency response
clg;
hold on;
N=[1 0 0 0 0];
D=[1 1.308E+4 8.534E+7 3.269E+11 6.250E+14];
bode(N,D);
end;
```

Matlab plot:

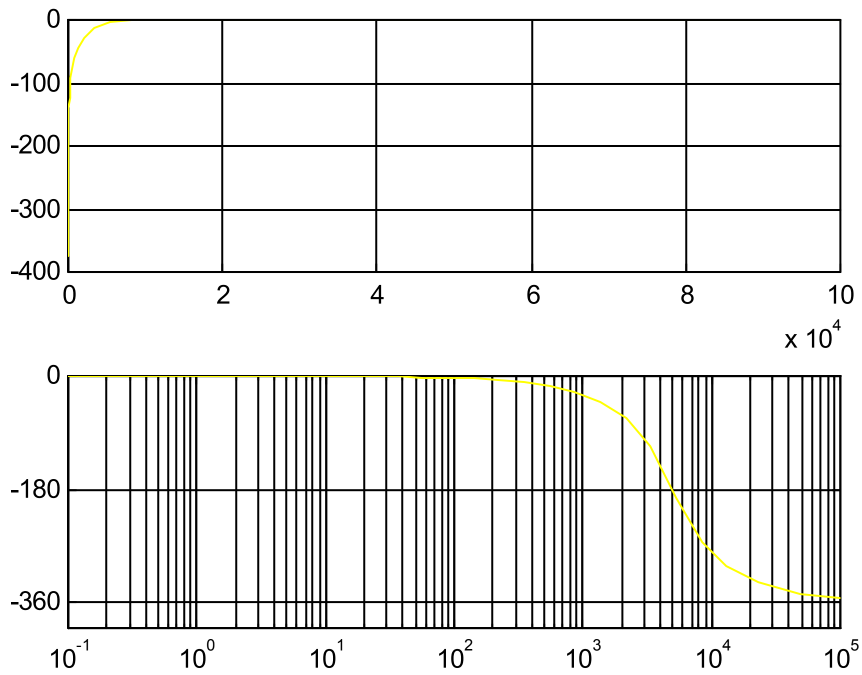

Figure 7 - Highpass Butterworth response

Highpass filter, Chebyshev approximation, ω=5000 rad/sec

Output:

```
R2= 1.613E+4  R4= 2.480E+4  R6= 4.587E+3  R8= 8.720E+4
```

Matlab function:

```
% Fourth-order Highpass filter frequency response
clg;
hold on;
N=[1 0 0 0 0];
D=[1 1.036E+4 6.850E+7 2.590E+11 6.250E+14];
bode(N,D);
end;
```

Matlab plot:

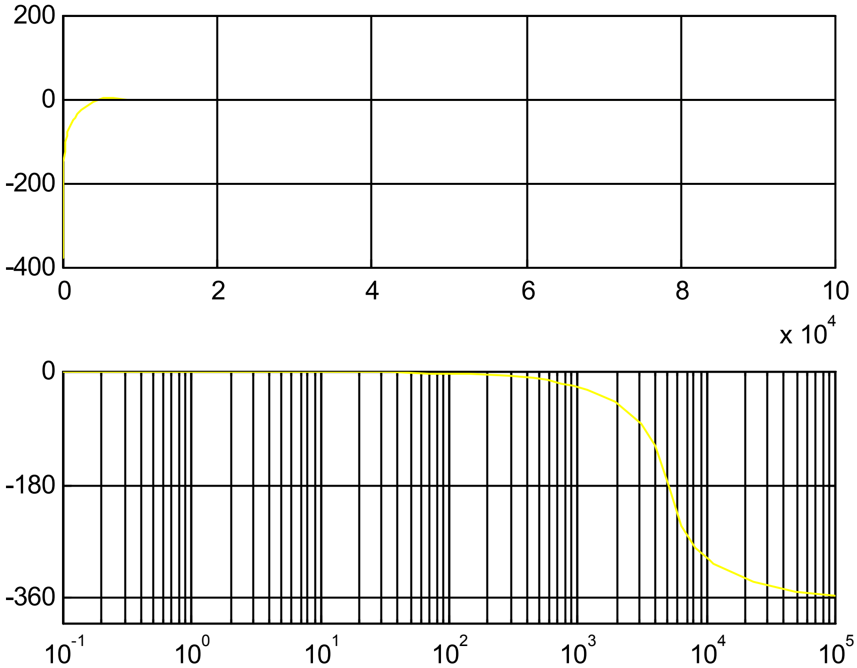

Figure 8 - Highpass Chebyshev response

Summary

Circuit designing systems have been in existence for some time. Each one was designed to serve a particular purpose or a certain field of industry. They all achieved some degree of success. This filter designer is reliable enough to give a reasonably accurate components for lowpass, highpass configurations using Butterworth and Chebyshev approximations.

Bibliography

R. Jensen, *Handbook of Circuit Analysis Languages and Techniques*, Prentice-Hall

J. Keown, *PSpice and Circuit Analysis*, Macmillan

R. Conant, *Engineering Circuit Analysis with PSpice Probe*, 1993, McGraw-Hill International

Appendices

REXX source code

The RXX file associated with the application is shown below. A table of the functions used to calculate the component values and the Matlab code is shown. They are found in the 'Global procedures' section below.

Table 2 - Module purposes

Function	*Purpose*
LowpassEval	Find lowpass filter components
HighassEval	Find highpass filter components
LowpassMakeMatlabFile	Write lowpass filter Matlab file
HighpassMakeMatlabFile	Write highpass filter Matlab file

```
/*------------------------------------------------------------------------+
|                                                                         |
|  REXX source code listing for DrRexx application:                       |
|    D:\Desktop\Work\Projects\FilD\FilDes1.RES                            |
|                                                                         |
|  File last modified on: 10/14/95 at: 02:58pm                            |
|  Listing produced on:   10/14/95 at: 03:04pm                            |
|                                                                         |
+------------------------------------------------------------------------*/

SIGNAL ON SYNTAX
SIGNAL ON HALT
SIGNAL INIT

RETURN:
  SIGNAL VALUE DrRexxEvent()
```

```
L1:
  EXIT -1

L2:
  INTERPRET DrRexxInterpret()
  SIGNAL RETURN

/*--------------------------------------------------------------------------------+
|                                                                                 |
|  Event handlers for dialog: MWin                                                |
|                                                                                 |
+--------------------------------------------------------------------------------*/

/* Event handlers for: MWin (DIALOG) */

MWin_Init:
  /* Loads the REXX MessageBox
     function used by WCornerFreq
     and MyButton */

  call RxFuncAdd 'RxMessageBox', 'RexxUtil', 'RxMessageBox'

  /* Stops the executable from
     initially flashing */

  call Show
  SIGNAL RETURN

/* Event handlers for: MatlabWin (MLE) */

MWin_MatlabWin_Init:
  /* Clear text window */

  call MatlabWin.Text ""

  /* Clear Matlab file */

  MatlabFile=""
  SIGNAL RETURN

/* Event handlers for: MyButton (PUSHBUTTON) */

MWin_MyButton_Click:
  /* Informs user about the author */

   TellMe=RxMessageBox('Filter designer version 1.1 - (C) 1995 by Antonino Iannella',
"About the author", "OK", "INFORMATION")
  SIGNAL RETURN

/* Event handlers for: ClipButton (PUSHBUTTON) */

MWin_ClipButton_Click:
  call clipboard MatlabWin.Text()
  SIGNAL RETURN

/* Event handlers for: ChooseMatlab (CHECKBOX) */

MWin_ChooseMatlab_Init:
  call ChooseMatlab.Select 0
  WriteMatlabFile='No'
  SIGNAL RETURN

MWin_ChooseMatlab_Click:
  /* Sets the WriteMatlabFile switch
     to create a Matlab file for the
     next filter calculated.  A Matlab
     file is output only after the
     Evaluate (FindResult) button
     is pressed.  */

  if ChooseMatlab.Select() = 1
     then
     WriteMatlabFile="Yes"
  else
     WriteMatlabFile="No"
  SIGNAL RETURN

/* Event handlers for: OutResult (ENTRYFIELD) */
```

```
MWin_OutResult_Init:
  NewResult=""
  SIGNAL RETURN

/* Event handlers for: FindResult (PUSHBUTTON) */

MWin_FindResult_Click:
  call ResetValues

  if FilterType="Lowpass"
     then
     do
       call LowpassEval
       if WriteMatlabFile="Yes"
          then
          call LowpassMakeMatlabFile
     end
  else
     do
       call HighpassEval
       if WriteMatlabFile="Yes"
          then
          call HighpassMakeMatlabFile
     end

  /* Write new components and Matlab file */

  call OutResult.Text NewResult
  SIGNAL RETURN

/* Event handlers for: WCornerFreq (ENTRYFIELD) */

MWin_WCornerFreq_Init:
  /* Initialise corner frequency */

  CornerFreq=WCornerFreq.Text()
  SIGNAL RETURN

MWin_WCornerFreq_LoseFocus:
  /* Check that the corner frequency
     is a valid number */

  if DATATYPE(WCornerFreq.Text()) = 'NUM'  /* Make sure it's a number */
     then
     do
       call WCornerFreq.Text ABS(WCornerFreq.Text())

       if WCornerFreq.Text()=0   /* Cannot be zero - issue warning */
          then
          do
                    Action=RxMessageBox('The  corner  frequency  must  not  be  zero!',
"Unsurmountable error", "OK", "WARNING")
            call WCornerFreq.Text CornerFreq
          end
       else
          CornerFreq=WCornerFreq.Text()  /* Accept new corner frequency */
     end
  else     /* Report error */
     do
       Action=RxMessageBox('The corner frequency must be a number!', "Fatal error",
"OK", "EXCLAMATION")
     call WCornerFreq.Text CornerFreq
     end
  SIGNAL RETURN

/* Event handlers for: ChebF (RADIOBUTTON) */

MWin_ChebF_Click:
  /* Set Chebyshev as default filter
     approximation by setting stage
     values q1, q2 according to
     polynomial */

  q1=0.62  /* Quality factors for Chebyshev filter */
  q2=2.18
  SIGNAL RETURN
```

```
MWin_ChebF_Init:
  call ChebF.Select 0
  SIGNAL RETURN

/* Event handlers for: ButtF (RADIOBUTTON) */

MWin_ButtF_Click:
  /* Set Butterworth as default filter
     approximation by setting stage
     values q1, q2 according to
     polynomial */

  q1=0.54  /* Quality factors for Butterworth filter */
  q2=1.31
  SIGNAL RETURN

MWin_ButtF_Init:
  /* Initialise Butterworth as the
     default filter */

  call ButtF.Select 1

  q1=0.54  /* Quality factors for Butterworth filter */
  q2=1.31
  SIGNAL RETURN

/* Event handlers for: Highp (RADIOBUTTON) */

MWin_Highp_Init:
  /* FilterType is set to Lowpass
     by default */

  call Highp.Select 0
  FilterType="Lowpass"
  SIGNAL RETURN

MWin_Highp_Click:
  /* Changes FilterType to Highpass */

  FilterType="Highpass"
  SIGNAL RETURN

/* Event handlers for: Lowp (RADIOBUTTON) */

MWin_Lowp_Init:
  /* FilterType is set to Lowpass
     by default */

  FilterType="Lowpass"
  SIGNAL RETURN

MWin_Lowp_Click:
  /* When either of these radio buttons
     are chosen, FilterType is modified
     to contain the selected filter */

  FilterType="Lowpass"
  call Lowp.Select 1
  SIGNAL RETURN

/*--------------------------------------------------------------------------------+
|                                                                                 |
|  Global procedures:                                                             |
|                                                                                 |
+--------------------------------------------------------------------------------*/

HighpassMakeMatlabFile:
  /* Forms the Matlab filter
     response function for the
     current filter.  This is
     only done when the Evaluate
     button is pressed.*/

  /* Find transfer function
     components separately */

  Hpart1=(2/c) * ((1/r4) + (1/r8))
  Hpart2=(1/(c*c)) * ((1/(r6*r8)) + (1/(r2*r4)) + (4/(r4*r8)))
```

```
  Hpart3=(2/(c*c*c*r4*r8)) * ((1/r6) + (1/r2))
  Hpart4=1/(c*c*c*c*r2*r4*r6*r8)

  /* Write the file */

  MatlabFile="% Fourth-order" FilterType "filter frequency response" d2c(13),
             "clg;" d2c(13) "hold on;" d2c(13),
             "N=[1 0 0 0 0];" d2c(13),
             "D=[1" format(Hpart1,,3,,0) format(Hpart2,,3,,0),
             format(Hpart3,,3,,0),
             format(Hpart4,,3,,0)"];" d2c(13),
             "bode(N,D);" d2c(13),
             "end;" d2c(13)

  call MatlabWin.Text MatlabFile
  RETURN

LowpassMakeMatlabFile:
  /* Forms the Matlab filter
     response function for the
     current filter.  This is
     only done when the Evaluate
     button is pressed.*/

  Lpart1=1/(r*r*r*r*c2*c4*c6*c8)
  Lpart2=(2/r) * ((1/c6) + (1/c2))
  Lpart3=(1/(r*r)) * ((1/(c6*c8)) + (1/(c2*c4)) + (4/(c2*c6)))
  Lpart4=(2/(r*r*r*c2*c6)) * ((1/c8) + (1/c4))

  /* Write the file */

  MatlabFile="% Fourth-order" FilterType "filter frequency response" d2c(13),
             "clg;" d2c(13) "hold on;" d2c(13),
             "N=["format(Lpart1,,3,,0)"];" d2c(13),
             "D=[1" format(Lpart2,,3,,0) format(Lpart3,,3,,0),
             format(Lpart4,,3,,0) format(Lpart1,,3,,0)"];" d2c(13),
             "bode(N,D);" d2c(13),
             "end;" d2c(13)

  call MatlabWin.Text MatlabFile
  RETURN

HighpassEval:
  /* Evaluates a highpass filter
     using current CornerFreq */

  c=10e-9   /* Capacitance values */

  r2=1/(2*q1*CornerFreq*c)  /* Stage 1 resistors */
  r4=(2*q1)/(CornerFreq*c)

  r6=1/(2*q2*CornerFreq*c)  /* Stage 2 resistors */
  r8=(2*q2)/(CornerFreq*c)

  NewResult="R2=" format(r2,,3,,0),
            " R4=" format(r4,,3,,0),
            " R6=" format(r6,,3,,0),
            " R8=" format(r8,,3,,0)
  RETURN

LowpassEval:
  /* Evaluates a lowpass filter
     using current CornerFreq */

  r=10e3   /* Resistance values */

  c2=(2*q1)/(CornerFreq*r)  /* Stage 1 capacitors */
  c4=1/(2*q1*CornerFreq*r)

  c6=(2*q2)/(CornerFreq*r)  /* Stage 2 capacitors */
  c8=1/(2*q2*CornerFreq*r)

  NewResult="C2=" format(c2,,3,,0),
            " C4=" format(c4,,3,,0),
            " C6=" format(c6,,3,,0),
            " C8=" format(c8,,3,,0)
  RETURN
```

```
ResetValues:
  /* Initialises all necessary
     variables to calculate the
     filter response */

  /* Clear text windows */

  call OutResult.Text ""
  call MatlabWin.Text ""

  /* Clear NewResult */
  NewResult=""
  RETURN

/*--------------------------------------------------------------------------------+
|                                                                                 |
|  Default initialization:                                                        |
|                                                                                 |
+--------------------------------------------------------------------------------*/

INIT:
  SIGNAL RETURN

/*--------------------------------------------------------------------------------+
|                                                                                 |
|  Default error handlers:                                                        |
|                                                                                 |
+--------------------------------------------------------------------------------*/

SYNTAX:
  SAY 'SYNTAX ERROR:' errortext( rc ) 'in:'
  SAY sourceline( sigl )
  SIGNAL ON SYNTAX
  SIGNAL RETURN

HALT:
  SAY 'HALT occurred in:'
  SAY sourceline( sigl )
  SIGNAL ON HALT
  SIGNAL RETURN
```

Index of figures and tables