RESEARCH ARTICLE OPEN ACCESS

Utilizing Symbolic Programming in Analog Circuit Synthesis of Arbitrary Rational Transfer Functions

Amjad Fuad Hajjar
Department of Electrical and Computer Engineering, King Abdulaziz University, Saudi Arabia

ABSTRACT
The employment of symbolic programming in analog circuit design for system interfaces is proposed. Given a rational transfer function with a set of specifications and constraints, one may autonomously synthesize it into an analog circuit. First, a classification of the target transfer function polynomials into 14 classes is performed. The classes include both stable and unstable functions as required. A symbolic exhaustive search algorithm based on a circuit configuration under investigation is then conducted where a polynomial in hand is to be identified. For illustration purposes, a set of complete design equations for the primary rational transfer functions is obtained targeting all classes of second order polynomials based on a proposed general circuit configuration. The design consists of a single active element and four different circuit structures. Finally, an illustrative example with full analysis and simulation is presented.

Keywords – Analog Circuits, Filters, Interface, Maple, Symbolic Programming, Transfer Functions

I. INTRODUCTION

Even with the advances in the digital world of electronics, analog circuit design is still the heart and most sensitive part of many engineering systems involving sensing and actuation. In consequence, multiple different ways are employed in designing analog circuits, hand-design being the obvious and foremost tactic. Nevertheless, automating the design process of analog circuits is becoming more crucial, not only for the time-to-market constraints but also for having better quality designs. Hence, automation methods are developed in this field such as heuristics inspection [1], knowledge-based synthesis [2], evolutionary computation involving genetic algorithms [3-7], and neural network based designs [8]. Once the synthesized circuit is completed, it is then verified to satisfy the required specifications.

In regard to the software applications assisting engineers in analog circuit design, a nice survey is presented in [9] comparing the different CAD tools in terms of features, simulation domains, speed, flexibility, and ease-of-use. Yet, most of the tools are based on numerical simulations and syntheses lacking the deployment of symbolic programming.

Computer Algebra Systems for symbolic and numeric programming have been developed for a long time and are becoming very power tools for researchers and engineers aiding in mathematical problems and simulations. When solving a problem by thoughts and ideas, it would be very helpful to get assistance from a tool that quickly carries out all the mathematical developments symbolically. Thus utilizing symbolic programming in this field is advantageous. The objective of this work is then to answer the following question: how to implement a certain transfer function in analog circuit to interface a specific unit in an electronic system?

In this paper, the utilization of symbolic programming in implementing arbitrary transfer functions to analog circuits is presented. First, a classification of an arbitrary polynomial is proposed, followed by an algorithm for symbolically and autonomously identifying the polynomial category. A general circuit configuration is also proposed as an example to emphasize the effectiveness of symbolic programming in this field. Finally a complete set of design equation for an arbitrary rational transfer function is illustrated.

II. POLYNOMIAL CATEGORIZATION

The focus of this work is to implement an arbitrary rational transfer function of the general form:

$$H(s)=\frac{a_0+a_1s+a_2s^2+\cdots+a_Ns^N}{b_0+b_1s+b_2s^2+\cdots+b_Ms^M} \qquad (1)$$

where a_i, b_j are real coefficients, a_N, $b_M \neq 0$, N and M are the orders of the numerator and denominator of the transfer function, respectively. Without loss of generality, studying up to the second order would be sufficient since the zeros and poles of the function must be real or complex conjugates, hence the higher orders can be obtained by cascading multiple lower order stages. In that, we classify the numerator and denominator polynomials into 14 categories based on their zeros as shown in Table I.

Table I. Polynomial Categories

#	Cat	Form	Zeros
01	*NZ*	a	-
02	*S0*	s	0
03	*S1N*	$s+a$	$-a$
04	*S1P*	$s-a$	$+a$
05	*D0*	s^2	0,0
06	*D1N*	$s.(s+a)$	$0,-a$
07	*D1P*	$s.(s-a)$	$0,+a$
11	*DQR*	s^2-a^2	$\pm a$
12	*DQC*	s^2+a^2	$\pm ja$
08	*DNN*	$(s+a).(s+b)$	$-a,-b$
09	*DNP*	$(s+a).(s-b)$	$-a,+b$
10	*DPP*	$(s-a).(s-b)$	$+a,+b$
13	*DXN*	s^2+as+b	$-A\pm jB$
14	*DXP*	s^2-as+b	$+A\pm jB$

Note: a, b, A, B are all positive constants

These polynomials can also be classified into two sets: *primary* and *secondary*, where the secondary polynomials can be obtained from the primary set using two cascaded stages. Thus, any arbitrary transfer function can be achieved when all the primary polynomials are implemented.

Table II. Primary and Secondary Polynomials

Primary Set		Secondary Set		
NZ	a	*D0*	s^2	*S0× S0*
S0	s	*D1N*	$s.(s+a)$	*S0× S1N*
S1N	$s+a$	*D1P*	$s.(s-a)$	*S0× S1P*
S1P	$s-a$	*DQR*	s^2-a^2	*S1N× S1P*
DQC	s^2+a^2	*DNN*	$(s+a).(s+b)$	*S1N× S1N*
DXN	s^2+as+b	*DNP*	$(s+a).(s-b)$	*S1N× S1P*
DXP	s^2-as+b	*DPP*	$(s-a).(s-b)$	*S1P× S1P*

III. CLASSIFICATION ALGORITHM

One may classify a polynomial expression simply by its order. However, the behavior differs among the same order based on the type of its zeros. For the first order polynomials, we certainly expect its zero to be real since we are implementing it with a real circuit. Thus, the classification of the first order polynomials is based on the sign of its zero. Let '*a'* be a positive real number. We then have three different classes (Table III):

Table III. 1st Order Zero Classes

Zero	Category	Form
0	*S0*	s
+ve	*S1P*	$s-a$
-ve	*S1N*	$s+a$

For a second order polynomial, its zeros can either be real or a pair of complex conjugates. In the real zero case, the nine categories shown in Table IV are possible based on the signs of the zeros, given that 'a' and 'b' are positive constants.

Table IV. 2nd Order Real Zero Classes

1st Zero	2nd Zero	Category	Form
0	0	*D0*	s^2
0	+ve	*D1P*	$s.(s-a)$
0	-ve	*D1N*	$s.(s+a)$
+ve	+ve	*DPP, DP2*	$(s-a).(s-b)$ or $(s-a)^2$
+ve	-ve	*DNP, DQR*	$(s+a).(s-b)$ or (s^2-a^2)
-ve	-ve	*DNN, DN2*	$(s+a).(s+b)$ or $(s+a)^2$

For the complex conjugate zeros, the real part of the zeros would distinguish the category of the polynomial as shown in Table V.

Table V. 2nd Order Complex Zero Classes

Real Part	Category	Form
0	*DQC*	(s^2-a^2)
+ve	*DXP*	s^2-as+b
-ve	*DXN*	s^2+as+b

Note that $4b > a^2$ in order for the polynomial to have complex zeros. Thus 'b' is always positive and the sign of 'a' would determine the category.

In classifying the category of a given polynomial expression, first we calculate its order. The Maple function `degree(expr,var)` gives the highest order of the provided expression with respect to the given variable. In our case, we look for the degrees 0, 1, and 2 only. However, the zeros of a given polynomial are hard to be distinguished whether they are positive or negative or even real or complex since they are symbolic expressions. Thus, the following algorithm is used:

0th Order: Zero degree expressions of the Laplace frequency, s, are the simplest. If required, a check whether the expression is positive or negative can be made by searching for a minus sign symbol. Since the investigated expressions are functions of resistors and capacitors, all additive and multiplicative combinations of components are positive unless there is a subtraction operation in the expression. In that case, one might be able to obtain a negative value by adjusting the components. If no minus symbol exits, then definitely the constant is positive. If however a minus sign is found, then there is a potential of having a negative constant. The Maple function '`search (string, pattern)`' is used to check the existence of the '`-`' symbol in an expression.

1st Order: For the first degree polynomials of the general form $a+bs$, we first evaluate the expression at $s=0$ using the Maple function '`eval (expr, var = value)`'. When the result is zero, then the expression is definitely classified as *S0* category disregarding any possible multiplicative constant; if not, then the expression has either a positive or a negative zero. When no minus symbol is detected in the expression (b/a) after simplification using the `simplify(expr)` function, then we classify the expression as *S1N* category; otherwise, it can potentially be an *S1P* category and will be labeled so. It is also possible to find one configuration that can generate both negative and positive zeros by setting the appropriate component values.

One may argue that we can get an *S0* class out of *S1N* or *S1P* by setting the constant a=0. It is a possibility; however, in most cases this causes a circuit violation, having impossible components values, or even zero valued components that eventually changes the circuit configuration. Mind the reader that once the negative feedback on an active op amp circuit is set open, the behavior of the circuit is no longer valid.

2nd Order: In classifying the second degree polynomials of the form $a+b.s+c.s^2$, we first extract the coefficients a, b, and c. The Maple function `coeff(expr,var,degree)` is used. Then, we check whether both (a=0 and b=0) to classify the *D0* class. If not, then when a=0 alone, then the category is either *D1P* or *D1N*. If however b=0 but $a\neq 0$, then we have one of the *DQs* categories. Finally, if none of the constants is zero, then we have one of the remaining *D2s* categories. To distinguish between (*D1P* and *D1N*) and (*DQC* and *DQR*), similar minus symbol checks to the *S1* case are made.

For the remaining cases of *D2*, we need to know whether the polynomial zeros are real or complex. Since the constants are symbolic, it is hard to recognize some of the categories definitely. We propose solving for the zeros symbolically first: if the simplified zeros do not contain a square root, then definitely the zeros are real due to the positive components values. The polynomial can then be classified as *DNN*, *DNP*, or *DPP*. We are still not sure whether the zeros are complex, but the expression will be labeled either *DXN* or *DXP* based on the existence of the minus symbol in b/c.

In summary, the above approach is used in classifying symbolic polynomials into the proposed categories. The conditions used in the classification algorithm when a target polynomial is sought are listed in Table VI, while the definitely and possibly classified polynomials are shown in Table VII; it also shows all the other possible categories in the indeterminate cases.

Table VI. Conditions for Polynomial Classification

Polynomial		***Conditions***
NZ	a	`degree(H,s)=0`
S0	s	`coeff(H,s,0)=0`
S1N	$s+a$	`coeff(H,s,1)/coeff(H,s,0)=1/a`
S1P	$s-a$	`coeff(H,s,1)/coeff(H,s,0)=-1/a`
D0	s^2	`coeff(H,s,0)=0,coeff(H,s,1)=0`
D1N	$s.(s+a)$	`coeff(H,s,2)/coeff(H,s,1)=1/a` `coeff(H,s,0)=0`
D1P	$s.(s-a)$	`coeff(H,s,2)/coeff(H,s,1)=-1/a` `coeff(H,s,0)=0`
DNN	$(s+a).(s+b)$	`p:=solve(H,s); p[1]=-a,p[2]=-b`
DNP	$(s+a).(s-b)$	`p:=solve(H,s); p[1]=-a,p[2]=b`
DPP	$(s-a).(s-b)$	`p:=solve(H,s); p[1]=a,p[2]=b`
DQR	s^2-a^2	`coeff(H,s,2)/coeff(H,s,0)=-1/a^2, coeff(H,s,1)=0`
DQC	s^2+a^2	`coeff(H,s,2)/coeff(H,s,0)=1/a^2` `coeff(H,s,1)=0`
DXN	$s^2+a.s+b$	`coeff(H,s,2)/coeff(H,s,0)=1/b` `coeff(H,s,1)/coeff(H,s,0)=1/a`
DXP	$s^2-a.s+b$	`coeff(H,s,2)/coeff(H,s,0)=1/b` `coeff(H,s,1)/coeff(H,s,0)=-1/a`

Table VII. Classification Possibilities

Classified as		***Possibilities are***
NZ	definitely	*NZ*
S0	definitely	*S0*
S1N	definitely	*S1N*
S1P	possibly	*S1N or S1P*
D0	definitely	*D0*
D1P	possibly	*D1N or D1P*
D1N	definitely	*D1N*
DPP	possibly	*DNN, DNP, DPP or DP2*
DNP	possibly	*DNN, DNP or DQR*
DNN	possibly	*DNN or DN2*
DQC	definitely	*DQC*
DQR	possibly	*DQR or DQC*
DXP	possibly	*DXP, DNN, DNP, DPP or DQC*
DXN	possibly	*DNN or DXN*

IV. CIRCUIT DESIGN CONFIGURATIONS

Arbitrary transfer functions can be implemented in different circuit configurations, both passive and active. For years, designers use well-known circuits developed long ago to implement their specific filters and functions with the modifications they introduced. For instant, a modified n^{th} order state variable filter is proposed in [10] using n integrators to implement LPF, HPF, and BPF filters. A study of the component sensitivity to the filter specifications is also presented. Another configuration specific to all-pole filters is shown in [11] to implement low-pass filters with very low sensitivity.

Although the goal of this work is to show the power of utilizing symbolic programming in achieving the desired interface, we propose a general circuit configuration based on one operational amplifier assumed to be ideal for simplicity. The technique would work for any other design configuration as well, and definitely engineers would prefer to develop their own ideas. The proposed circuit configuration is shown in Fig. 1.

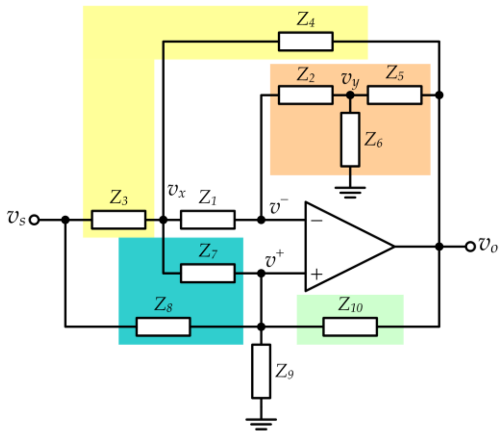


Fig. 1. The proposed general circuit configuration

The four shaded networks in the general design can be utilized to give more options to the resultant circuit gain, or transfer function. If one of the networks is not needed, it is then replaced suitably. The main benefit of each network follows:

1. **(TS) T-Structure Feedback Network**: considered by the impedance blocks Z_2, Z_5, and Z_6. It adds possibilities for complex poles and zeros to the gain function.
2. **(DF) Double Feedback Network**: characterized by the impedances Z_3 and Z_4. The only possible network to yield double differentiator circuits.
3. **(DI) Differential Input Network**: characterized by the impedance blocks Z_7 and Z_8. The network yields positive zeros and poles.
4. **(PF) Positive Feedback Network**: characterized by the impedance blocks Z_9 and Z_{10}. It gives the flexibility of having both positive and negative zeros and poles.

The general circuit can be configured by utilizing as many networks as possible. When a network is not utilized, it will be eliminated properly (e.g. the *T-Structure* is replaced by impedance Z_2 only, shorting Z_5 and opening Z_6 out). In that, the total number of the different circuit configurations is 16 as shown in Fig. 3. Table VIII summarizes the different circuit configurations setup and the number of remaining impedance blocks.

In regard to the impedance configuration, many arrangements are possible to include passive or even active components. In this work, we propose only the use of passive resistors and capacitors as they are the most used elements in analog circuits. The arrangements used are only four types shown in Fig. 2, namely a resistor, a capacitor, a series resistor with a capacitor, and a parallel resistor with a capacitor.

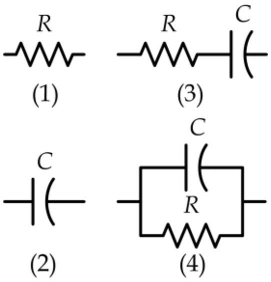


Fig. 2. The proposed impedance configurations

Table VIII. Circuit Configurations Setup

cfg.	*TS*	*DI*	*PF*	*DF*	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}	*N*
A00	0	0	0	0	.	.	0	∞	0	∞	∞	∞	0	∞	2
A01	0	0	0	1	.	.	.	.	0	∞	∞	∞	0	∞	4
A02	0	0	1	0	.	.	0	∞	0	∞	∞	∞	.	.	4
A03	0	0	1	1	.	.	.	.	0	∞	∞	∞	.	.	6
A04	0	1	0	0	.	.	0	∞	0	∞	∞	.	.	∞	4
A05	0	1	0	1	.	.	.	.	0	∞	.	.	.	∞	7
A06	0	1	1	0	.	.	0	∞	0	∞	∞	.	.	.	5
A07	0	1	1	1	.	.	.	.	0	∞	.	.	.	.	8
A08	1	0	0	0	.	.	0	∞	.	.	∞	∞	0	∞	4
A09	1	0	0	1	.	.	.	.	.	.	∞	∞	0	∞	6
A10	1	0	1	0	.	.	0	∞	.	.	∞	∞	.	.	6
A11	1	0	1	1	.	.	.	.	.	.	∞	∞	.	.	8
A12	1	1	0	0	.	.	0	∞	.	.	∞	.	.	∞	6
A13	1	1	0	1	.	.	.	.	.	.	.	.	.	∞	9
A14	1	1	1	0	.	.	0	∞	.	.	∞	.	.	.	8
A15	1	1	1	1	.	.	.	.	.	.	.	.	.	.	10

N: number of impedance blocks exists in the configuration

V. Circuit Analysis

Assuming ideal operational amplifiers and using Kirchhoff laws, one could analyze each of the 16 different circuits of Table VIII and calculate the gain as a function of all circuit impedances. Alternatively, we could analyze the general configuration and then substitute for the short or open circuits for the missing impedance blocks. The circuit analysis we present utilizes the symbolic programming too. Starting with Kirchhoff Current Law at the different nodes of the circuit and enforcing the equality of the positive and negative voltage terminals of the op amp, we have the set of equations in (2).

$$\begin{cases} \dfrac{v_s - v_x}{Z_3} = \dfrac{v_x - v_o}{Z_4} + \dfrac{v_x - v^-}{Z_1} + \dfrac{v_x - v^+}{Z_7} \\ \dfrac{v_x - v^+}{Z_7} + \dfrac{v_s - v^+}{Z_8} = \dfrac{v^+}{Z_9} + \dfrac{v^+ - v_o}{Z_{10}} \\ \dfrac{v_x - v^-}{Z_1} = \dfrac{v^- - v_y}{Z_2} \\ \dfrac{v^- - v_y}{Z_2} = \dfrac{v_y}{Z_6} + \dfrac{v_y - v_o}{Z_5} \\ v^+ = v^- \end{cases} \tag{2}$$

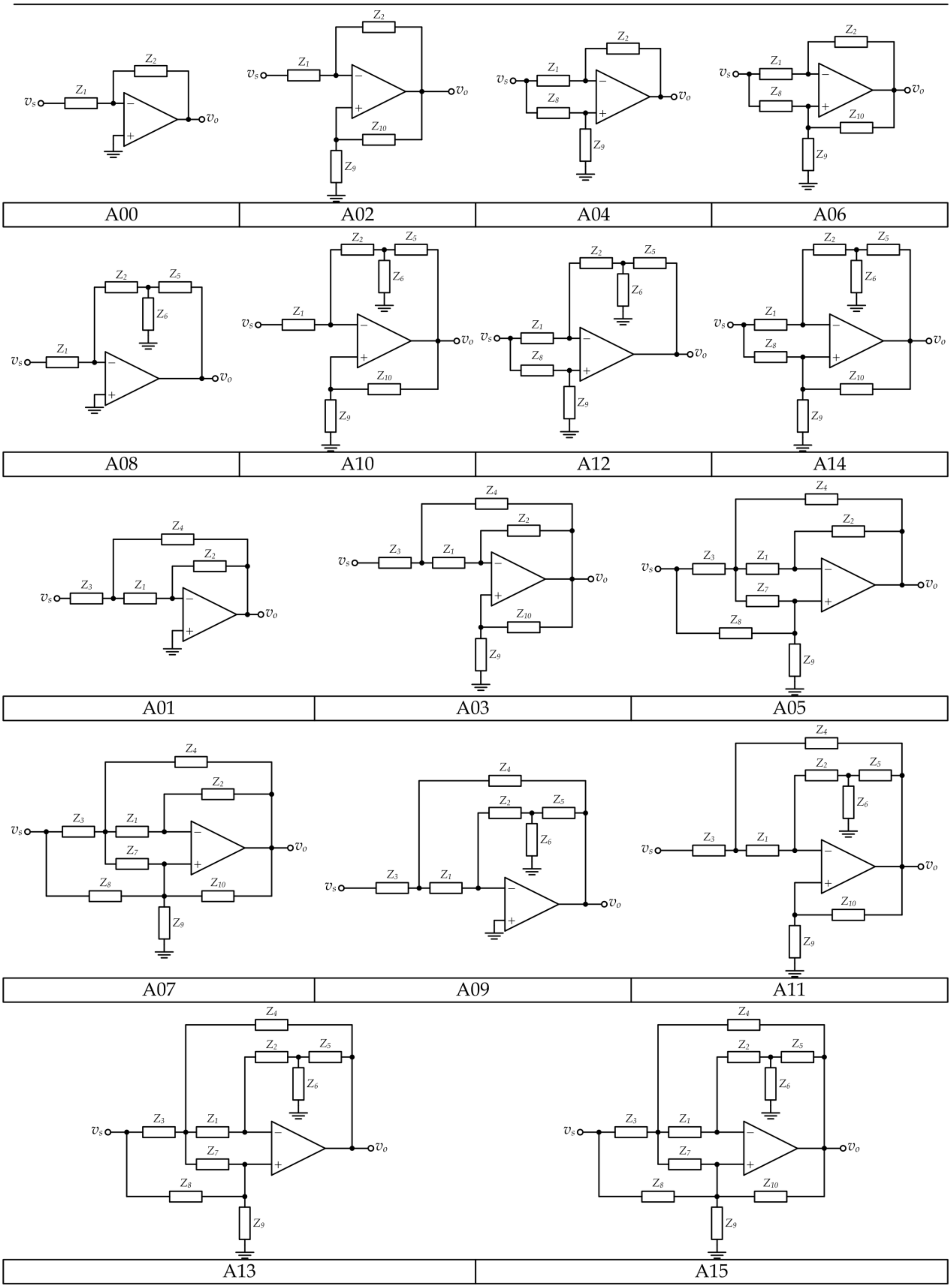


Fig. 3. Possible circuit configurations

We solve these five equations with respect to the five unknowns v_x, v_y, v^+, v^-, and v_o, using the Maple function `solve({eqns},{var})` to get the solution for v_o, which is quite long and includes 40 additions, 210 multiplications, and one division. The software package also includes an optimization library to reduce the number of arithmetic operations by introducing temporary variable assignments: `optimize(expr)`. When utilized, the optimum circuit transfer function now contains only 29 additions, 45 multiplications, one division, and 15 assignments. The optimized circuit transfer function assignments are listed in Table IX.

Once the general circuit is analyzed, the transfer functions of the different configurations can then be obtained by taking the limit as their specific impedances reach either zero or infinity according to Table VIII. For example, the circuit configuration A04 has a transfer function calculated by the Maple command `limit (H,{Z3=0,Z4=infinity, Z5=0,Z6=infinity,Z7=infinity, Z10= infinity})`. It is then further optimized to have a minimum number of arithmetic operations.

Table IX. Optimized General Transfer Function

```
t1  = Z2+Z5
t2  = Z2*Z6
t3  = Z5*Z9
t4  = Z7*Z9
t5  = -Z1-Z4
t6  = -Z2-Z6
t7  = Z6*Z10*Z1
t8  = Z10*t4
t9  = t6*Z8
t10 = Z6*t8
t11 = Z3*Z5*t8
t12 = Z4*Z9*t7
t13 = Z4*t10
t14 = Z1*t13+Z2*t11+Z8*t12+(t12+t13+
     (Z1+t1)*t10)*Z3

H   = (Z1*t11+((-Z10-Z9)*Z8*Z7*t2+(t9*
      t4+((Z3+Z8)*Z9*Z1+((Z3+Z1)*Z9+t9)
      *Z7)*Z10)*Z5)*Z4+t14)/(((-Z4*t3+(
      -t3+Z6*Z4)*Z10)*Z3*Z1+((t7+(-t2+(
      -Z1+t6)*Z5)*Z9)*Z4+(t5*t3+(Z2*Z5+
      (-t5+t1)*Z6)*Z10)*Z3)*Z7)*Z8+t14)
```

VI. TRANSFER FUNCTION SEARCH

Among the 16 different possible circuit configurations, A00 to A15, an exhaustive search begins incorporating the 4 suggested configurations for each impedance block in every circuit. In each search, the definite classifications of the numerator and denominator of the circuit transfer function are reported in a database. Mind the reader that the algorithm gives definite classifications as well as possibilities; hence, extra tests are performed later for those cases.

First we report the statistics of the classification of each configuration. Table X shows the number of polynomials in either the numerator or the denominator of the transfer function classified by the algorithm as such. Of course a better statistics would be to distinguish the rational function in detail, but this would make the table size 14^2 = 196 (column) for each of the 16 configuration (row).

The numbers in Table X give an idea on how the different circuit configurations yield the different polynomial classes. For instant, none of the configurations yields a *DQC* or a *DQR* polynomial in definite. On the other hand we noticed that the most polynomials found were *DXPs* and *DXNs* which could possibility turn out to *DQCs* or *DQRs*.

VII. RESULTS

A. The Primary Set Coverage

The minimum set of transfer functions that can implement any given function is the set of the rational functions with prime polynomials; namely the 13 transfer functions shown in the first row of Table XI. Mixed transfer functions or higher order functions can simply be built by cascading multiple primes. Table XI shows also the coverage count of each circuit configuration to each prime transfer function at the first search phase; i.e., when considering the define detections. Again the definitely detected polynomials by the algorithm which are also prime ones are only four: *NZ*, *S0*, *S1N*, and *DQC*. These are highlighted in the table. The remaining primes (*S1P*, *DXN*, and *DXP*) were extracted from the second search phase of possible detections. Notice that the transfer functions involving *DXP* were never detected in the definite phase.

Once all the prime transfer functions were extracted, selection criteria were applied to choose the optimum circuit implementation for a given transfer function. We consider in this work the following preferable features:

- **Real Components**: obviously implementations with negative-valued components are rejected, which can be symbolically tested case by case.
- **Component Count**: the least number of components, especially the capacitors are more preferable.
- **Feature Independence**: the wider constants ranges such as gain, frequency, quality, and damping coefficient which impose least restrictions are selected.
- **Identical Elements**: implementations with as many equal capacitors and resistors values are more preferable than the mixed ones.

Table X. Circuit Configurations Setup

cfg.	*NZ*	*S0*	*S1N*	*S1P*	*D0*	*D1P*	*D1N*	*DPP*	*DNP*	*DNN*	*DQC*	*DQR*	*DXP*	*DXN*
A00	10	8	12	0	0	0	0	0	0	2	0	0	0	0
A01	23	37	98	0	8	44	0	0	0	33	0	0	131	0
A02	15	9	84	72	0	30	8	2	63	74	0	0	29	48
A03	46	74	395	171	16	273	5	0	877	474	0	0	127	194
A04	15	9	84	72	0	30	8	2	63	74	0	0	29	48
A05	4	0	148	192	0	20	64	0	1226	0	0	0	1765	217
A06	10	2	0	262	0	0	42	4	547	0	0	0	68	241
A07	4	0	0	671	0	0	163	0	6101	0	0	0	423	1482
A08	29	36	131	0	9	55	0	0	10	71	0	0	63	0
A09	20	26	438	0	6	242	0	0	0	300	0	0	940	0
A10	12	4	170	138	0	78	38	0	601	568	0	0	182	121
A11	40	52	587	174	12	455	2	0	2620	1434	0	0	236	12
A12	47	75	354	152	16	256	12	2	881	410	0	0	30	109
A13	4	0	0	490	0	0	150	0	6114	0	0	0	355	611
A14	9	1	0	443	0	0	121	2	2713	0	0	0	99	560
A15	4	0	0	628	0	0	212	0	11760	0	0	0	70	990
Sum	**292**	**333**	**2501**	**3465**	**67**	**1483**	**825**	**12**	**33576**	**3440**	**0**	**0**	**4547**	**4633**

Table XI. Phase I Search Detection of the Prime Transfer Function Set

TF :	1 $\frac{NZ}{NZ}$	2 $\frac{NZ}{S0}$	3 $\frac{NZ}{S1N}$	4 $\frac{NZ}{S1P}$	5 $\frac{NZ}{DQC}$	6 $\frac{NZ}{DXN}$	7 $\frac{NZ}{DXP}$	8 $\frac{S0}{NZ}$	9 $\frac{S1N}{NZ}$	10 $\frac{S1P}{NZ}$	11 $\frac{DQC}{NZ}$	12 $\frac{DXN}{NZ}$	13 $\frac{DXP}{NZ}$
A00	2	1	2	-	-	-	-	1	2	-	-	-	-
A01	2	-	11	-	-	7	-	-	-	-	-	-	-
A02	4	-	-	6	-	-	-	-	1	-	-	-	-
A03	4	-	-	22	-	16	-	-	-	-	-	-	-
A04	4	-	1	-	-	-	-	-	-	6	-	-	-
A05	2	-	-	-	-	-	-	-	-	-	-	-	-
A06	4	-	-	1	-	-	-	-	-	1	-	-	-
A07	2	-	-	-	-	-	-	-	-	-	-	-	-
A08	2	1	1	-	-	-	-	1	13	-	-	5	-
A09	2	-	10	-	-	5	-	-	-	-	-	-	-
A10	4	-	-	4	-	-	-	-	-	-	-	-	-
A11	4	-	-	20	12	-	-	-	-	-	-	-	-
A12	4	-	1	-	-	-	-	-	-	22	16	-	-
A13	2	-	-	-	-	-	-	-	-	-	-	-	-
A14	4	-	-	1	-	-	-	-	-	-	-	-	-
A15	2	-	-	-	-	-	-	-	-	-	-	-	-
Sum	**48**	**2**	**26**	**54**	**12**	**28**	**0**	**2**	**16**	**29**	**16**	**5**	**0**

Based on these suggested criteria, Table XIII shows a solution to the prime transfer function set along with the design equations for a given feature set. This specific list however may not be preferable in different situations due to other selection criteria than the ones chosen in this study. For example, some applications have more emphasis on the input impedance, power ratings, noise levels, error offsets, and so on. For that, the readers have the freedom to program their own desired set of preferences that best suit their applications.

In general, the circuit configurations A00, A02, A04, A11, and A12 implement the whole prime transfer function set. Note that the second order polynomials of the prime set are implemented with similar configurations as possible by adjusting the 'd' parameter in the range $-1 < d < +1$, including the case $d = 0$ for the *DQC* polynomials.

For the last entry of Table XIII, the design equations use a constant α, which is the real solution of a 3rd order equation. Fig. 4 shows this α value at different cases.

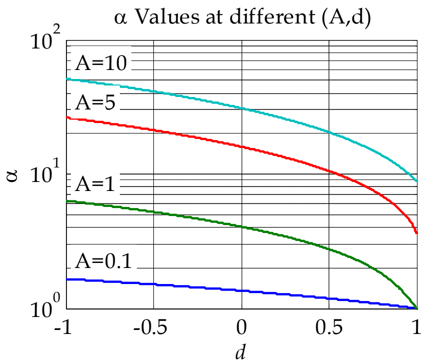


Fig. 4. α values at different gain and d constants

B. Secondary Set Coverage

Although any transfer function can be implemented by decomposing it into cascaded prime functions, the proposed circuit configuration is capable of implementing higher order functions directly. First, it is worth mentioning that some of configurations can uniquely implement certain secondary transfer functions that no other configuration is capable of. Table XII shows these configurations and their unique secondary transfer function set.

Table XII. Unique Coverage of Configurations

cfg.	*TF*	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}
A00	S0/DNN	3	4	0	∞	0	∞	∞	∞	0	∞
A02	S1N/DQR	1	2	0	∞	0	∞	∞	∞	2	1
	D1N/DQR	2	1	0	∞	0	∞	∞	∞	1	2
A06	DXN/DQR	1	3	0	∞	0	∞	∞	1	2	1
A08	D1N/NZ	2	1	0	∞	1	2	∞	∞	0	∞
	DNN/NZ	2	3	0	∞	1	2	∞	∞	0	∞
	S1N/D0	1	2	0	∞	2	1	∞	∞	0	∞
	DNN/D0	4	2	0	∞	2	1	∞	∞	0	∞
	S1N/DNN	1	4	0	∞	4	3	∞	∞	0	∞
	D1N/DNN	2	4	0	∞	4	1	∞	∞	0	∞
	DNN/DNN	4	4	0	∞	4	1	∞	∞	0	∞
A11	NZ/DQC	1	2	1	2	2	2	∞	∞	1	1
	D0/DQC	3	1	3	1	1	1	∞	∞	1	1

0: short circuit, ∞: open circuit, 1:4 according to Fig. 2

The reminder of all possible second order transfer functions can be tried out too and with multiple possible solutions. In this paper, the prime set of transfer functions is very sufficient to demonstration the usefulness of utilizing the symbolic programming in implementing a given transfer function into analog circuit. An example however is illustrated; suppose that the following differentiator interface function given in (3) is needed to be implemented in an analog active circuit:

$$H = \frac{100s}{10{,}000 + 100s + s^2} \tag{3}$$

The transfer function in (3) has two complex conjugate poles with negative real parts, and one zero at zero. It is then classified as *S0/DXN*, which can either be decomposed into two prime functions *S0/NZ* and *NZ/DXN* or can directly be implemented by searching for the suitable circuit configuration. The search in the constructed database of the general configuration circuit yields 42 potential solutions involving A01 and A09 circuit configurations. When examining the possible solutions, some were found to implement the given function but with negative component values, while others have more component count. The programmed conditions when a configuration is used from the database are:

- `coeff(denom(H),s,1)/coeff(denom(H),s,0)=2*d/w0`
- `coeff(denom(H),s,2)/coeff(denom(H),s,0)=1/w0^2`

When executed in Maple, the design equations for the circuit component are:

$$\begin{aligned} R_1 &= R_4 \frac{1}{2dR_4C_3\omega_0 - 1} \\ R_4 &= R_4 \\ C_2 &= \frac{2dR_4C_3\omega_0 - 1}{R_4^2C_3\omega_0^2} \\ C_3 &= C_3 \end{aligned} \tag{4}$$

Choices for R_4 and C_3 in (4) are open as long as all the other components become positive. We can then choose to set R_4 to resonate C_3 and the design equations of (4) become:

$$R_1 = \frac{1}{d\omega_0C_0} \quad R_4 = \frac{1}{d\omega_0C_0} \quad C_2 = d^2C_0 \quad C_3 = C_0 \tag{5}$$

Finally, for the transfer function of (3), the actual circuit components for a given choice of C_0 are:

$$R_1 = 20\text{k}\Omega \quad R_4 = 20\text{k}\Omega \quad C_2 = 250\text{nF} \quad C_3 = 1\mu\text{F} \tag{6}$$

And the circuit implementation is shown in Fig. 5.

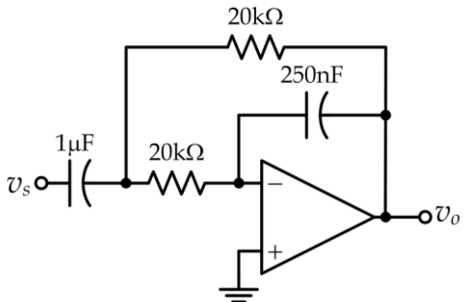
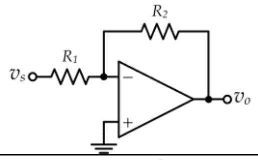
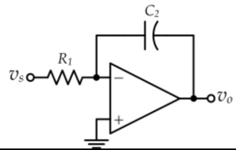
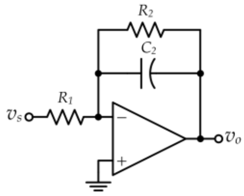
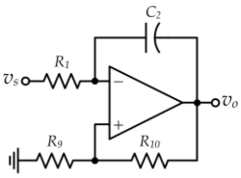
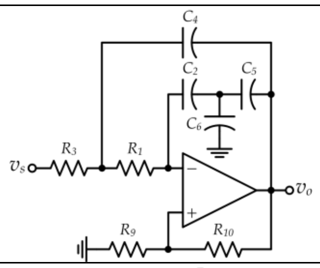
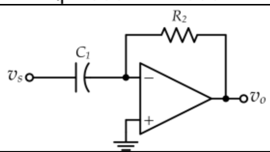
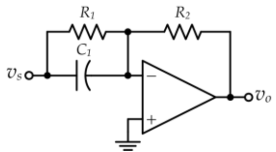
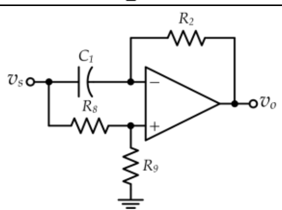
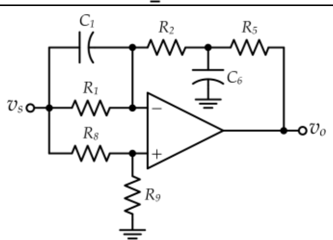
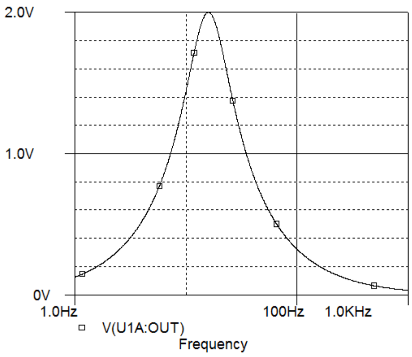


Fig. 5. Circuit implementation of example transfer function (3)

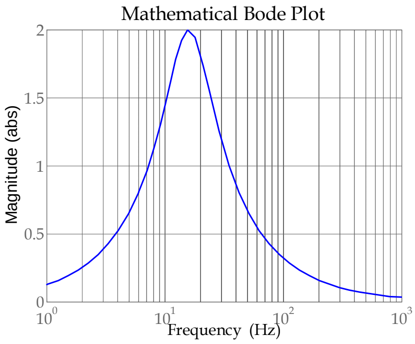
Circuit simulation is carried out to prove the equivalence of the transfer function. Fig. 6 shows the bode diagrams from the circuit simulator and the mathematical transfer function, which proves the equivalence. Note that the choices for the independent variables make room for other considerations such as the gain of the circuit. In this specific example, the gain of the transfer function and the circuit match with the selected values.

Table XIII. Optimum circuit implementation of the prime transfer function set

$\frac{NZ}{NZ}$	$H=-A$		$R_1 = R$ $R_2 = A\cdot R$
$\frac{NZ}{S0}$	$H=-\frac{A}{s}$		$R_1 = \frac{1}{A\cdot C}$ $C_2 = C$
$\frac{NZ}{S1N}$	$H=-\frac{A\omega_0}{s+\omega_0}$		$R_1 = \frac{1}{A\omega_0 C}$ $R_2 = \frac{1}{\omega_0 C}$ $C_2 = C$
$\frac{NZ}{S1P}$	$H=-\frac{A\omega_0}{s-\omega_0}$		$R_1 = \frac{1}{\omega_0 C\cdot(A-1)}$ $R_9 = R$ $R_{10} = (A-1)\cdot R$ $C_2 = C$
$\frac{NZ}{DQC}$ $\frac{NZ}{DXN}$ $\frac{NZ}{DXP}$	$H=\frac{A\omega_0^2}{s^2+2d\omega_0 s+\omega_0^2}$ $1.25< A <2$ $-1< d <1$		$R_1 = \frac{-3d+\sqrt{9d^2+12A-15}}{\omega_0 C\cdot(A-2)}$ $R_3 = \frac{3d+\sqrt{9d^2+12A-15}}{\omega_0 C\cdot(5-4A)}$ $R_{10} = (A-1)R$ $R_9 = R$ $C_2 = C_4 = C_5 = C_6 = C$
$\frac{S0}{NZ}$	$H=-A\cdot s$		$R_2 = \frac{A}{C}$ $C_1 = C$
$\frac{S1N}{NZ}$	$H=-\frac{A}{\omega_0}\cdot(s+\omega_0)$		$R_1 = \frac{1}{\omega_0 C}$ $R_2 = \frac{A}{\omega_0 C}$ $C_1 = C$
$\frac{S1P}{NZ}$	$H=-\frac{A}{\omega_0}\cdot(s-\omega_0)$ $A<1$		$R_2 = \frac{A}{\omega_0 C\cdot(1-A)}$ $R_9 = \frac{A}{1-A}R$ $R_8 = R$ $C_1 = C$
$\frac{DQC}{NZ}$ $\frac{DXN}{NZ}$ $\frac{DXP}{NZ}$	$H=-\frac{A}{\omega_0^2}\left(s^2+2d\omega_0 s+\omega_0^2\right)$ $-1<d<1$ where α is the real solution of: $\alpha^3-[1+A(3-2d)]\alpha^2+2dA\alpha-A$		$R_1 = 1/\omega_0 C$ $R_2 = \frac{\alpha(\alpha-1)}{\omega_0 C(\alpha^2-2d\alpha+1)}$ $R_5 = \alpha/\omega_0 C$ $R_9 = \frac{2\alpha^2(1-\alpha)}{\alpha^2-2d\alpha+1}R$ $R_8 = R \quad C_1 = C_6 = C$



(a)



(b)

Fig. 6. Bode plots of (a) circuit simulator, and (b) the mathematical transfer function

VIII. Conclusion

Utilizing symbolic programming in autonomous circuit synthesis was presented. Although a general circuit configuration was proposed to implement any arbitrary transfer function, the goal of this work was to show the usefulness of symbolic programming for engineers and researchers in advancing their new ideas and designs whenever needed. The 2nd order transfer function study presented can be expanded intelligently to accommodate for higher order functions as well as better designs suitable for ones needed and requirements.

References

[1] G. Sussman, R. Stallman, "Heuristic Techniques in Computer-Aided Circuit Analysis," *IEEE Transaction on Circuits and Systems*, CAS-22, 1975

[2] R. Harjani, R. Rutenbar, L. Carey, "A Prototype Framework for Knowledge-Based Analog Circuit Synthesis," *24th Design Automation Conference*, 1987, 42–49

[3] J. Lohn, S. Colombano, "A Circuit Representation Technique for Automated Circuit Design," *IEEE Transactions on Evolutionary Computation*, 3(3), Sept. 1999

[4] J. Lohn, G. Haith, S. Colombano, D. Stassinopoulos, "Towards Evolving Electronic Circuits for Autonomous Space Applications," *IEEE Aerospace Conference Proceedings*, 5, March 2000, 473-486

[5] S. Zhao, L. Jiao, J. Zhao, Y. Wang, "Evolutionary Design of Analog Circuits with a Uniform-Design Based Multi-Objective Adaptive Genetic Algorithm," *IEEE NASA/DoD Conference on Evolvable Hardware*, July 2005, 26-29

[6] A. Das, R. Vemuri, "GAPSYS: A GA-based Tool for Automated Passive Analog Circuit Synthesis," *IEEE International Symposium on Circuits and Systems*, May 2007, 2702-2705

[7] J. He, K. Zou, M. Liu, "Section-Representation Scheme for Evolutionary Analog Filter Synthesis and Fault Tolerance Design," *3rd Int'l Workshop on Advanced Computational Intelligence*, August 2010, 265-270

[8] F. Wang, Y. Li, "Analog Circuit Design Automation Using Neural Network-Based Two-Level Genetic Programming," *5th International Conference on Machine Learning and Cybernetics*, Dalian, August 2006, 2087-2092

[9] R. Henderson, L. Ping, J. Seweil, "Analog Integrated Filter Compilation," *Analog Integrated Circuits and Signal Processing*, 3, 1993, 217-228

[10] A. Oksasoglu, L. Huelsman, "A Modified State Variable Filter Realization," *Analog Integrated Circuits and Signal Processing*, 1, 1991, 21-32

[11] G. Moschytz, "Low-Sensitivity, Low-Power Active-RC Allpole Filters Using Impedance Tapering," *IEEE Transactions on Circuits And Systems—II: Analog and Digital Signal Processing*, 46(8), 1999