УДК 621.372.51

Особенности синтеза лестничных трансформирующих фильтров высокого порядка

А.Е. Курочкин, email: kurochkin@bsuir.by
Белорусский государственный университет информатики и радиоэлектроники,
Республика Беларусь, 220013, г. Минск, ул. П. Бровки, д.6

Аннотация. Рассмотрены особенности синтеза лестничных трансформирующих фильтров нижних частот высокого порядка с частотной характеристикой Чебышёва по методу Кауэра в виде первой канонической схемы. Показано, что использование стандартной компьютерной математики приводит к значительному росту погрешности при согласовании нагрузок фильтрами высокого порядка. Даны оценка погрешности расчёта нормированных параметров и рекомендации по применению математики произвольной точности

Ключевые слова: согласование сопротивлений, трансформирующий фильтр, транспонирование частоты, аппроксимация, полином Чебышёва первого рода, метод Кауэра, лестничная цепь, математика произвольной точности

Abstract. The features of the synthesis of high-order ladder transforming low-pass filters with the Chebyshev frequency response by the Kawer method in the form of the first canonical scheme are considered. It is shown that the use of standard computer mathematics leads to a significant increase in the error when matching loads of high order filters. An estimate of the error in calculating the normalized parameters and recommendations for the use of arbitrary precision mathematics are given

Keywords: impedance matching, transforming low pass filter, transposing of frequency, approximation, Chebyshev polynomials of the first kind, Cauer synthesis procedure, ladder circuit, mathematics of multiple precision

1. Введение

С точки зрения инженерной практики процедуре синтеза трансформирующих лестничных фильтров нижних частот (ФНЧ) (рис. 1) и вопросам практического согласования с их помощью неравных нагрузок в технической литературе уделяется внимание.

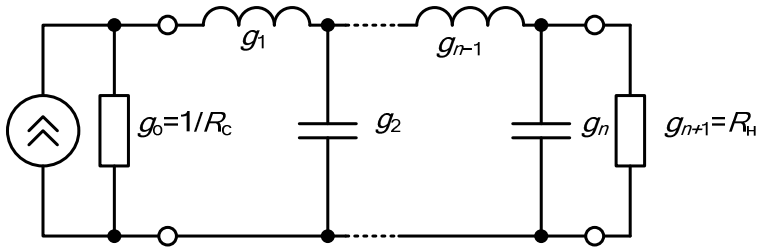


Рис. 1. Лестничный ФНЧ

Таблицы с нормированными параметрами элементов согласующих фильтров до 10 порядка предлагаются в [1] для относительных полос от 0,1 до 1 с шагом 0,1 и дискретных значений коэффициентов трансформации до 50. Но дискретность исходных параметров не позволяет считать методику универсальной. К тому же, как оказывается, табулированные данные не всегда удовлетворяют требуемой погрешности расчёта. Например, в таблице 1 представлены параметры g_k из [1], где k – номер элемента фильтра от $k = 0$ до $k = 11$, характеризующие нормированные значения индуктивности последовательных или ёмкости параллельных ветвей согласующего фильтра 10-го порядка для $r = R_н/R_c = 50$ в нормированной полосе $W = 0{,}3$. А на рис. 2 и рис.3 представлены результаты проверочного моделирования этого фильтра в программе Filter Solutions 8.0 [1].

Таблица 1**.** Значения нормированных параметров фильтра лестничной структуры для n=10

Нормированные параметры g_k											
g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}
1	**0.789986**	**0.846863**	**2.59137**	**0.504294**	**7.89711**	**0.157942**	**25.2147**	**0.051827**	**42.34315**	**0.0157997**	**50**

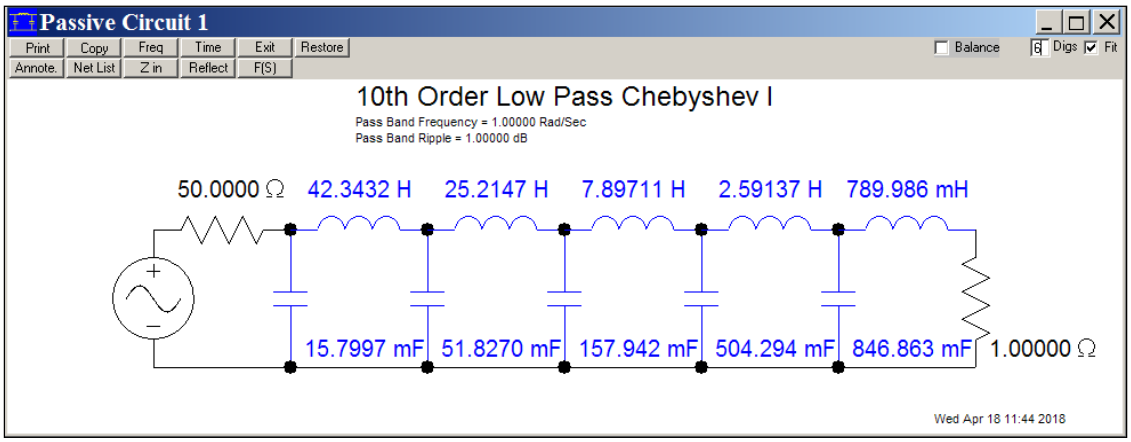


Рис.2. Схема проверяемого фильтра в программе Filter Solutions

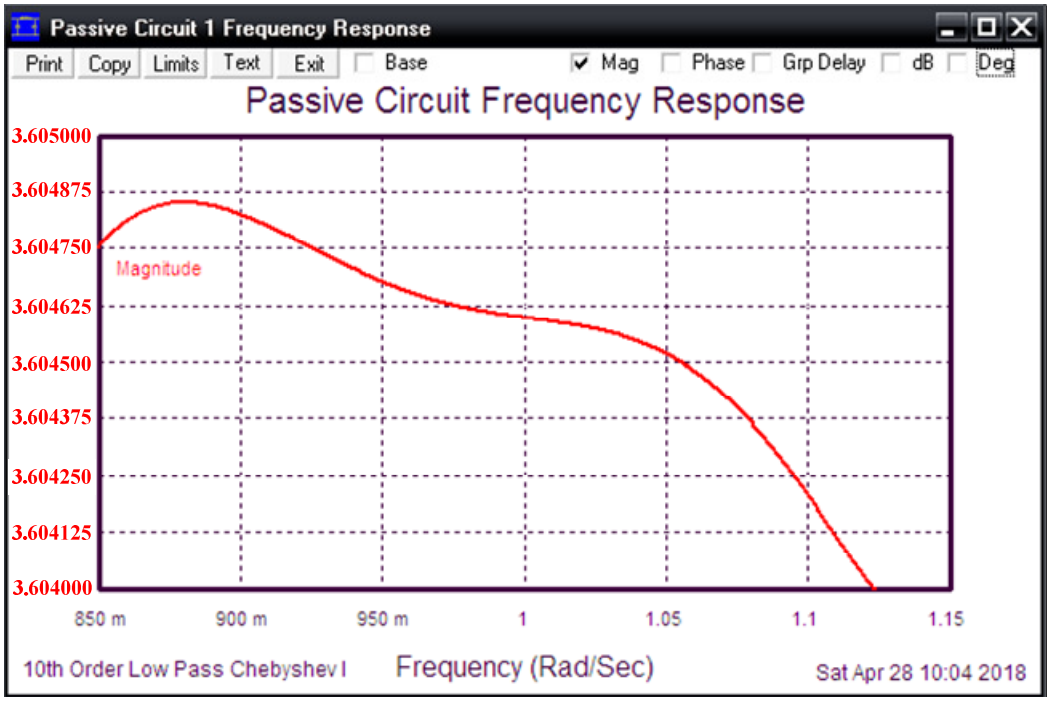


Рис. 3. График амплитудно-частотной характеристики (АЧХ) проверяемого фильтра

Как видно из рис. 3 график не соответствует обещанной равноволновой аппроксимации по Чебышёву. Многократное моделирование подтверждает, что табулированные данные в [2] для фильтров, начиная с 8-го порядка, не следует считать достоверными! В [3], [4] и [5] предлагаются громоздкие выражения для расчёта согласующих фильтров только до 10-14 порядков. В [4] приводятся

листинги компьютерных программ на языке Фортран и пример расчёта структуры не выше 16-го порядка. Таким образом, возникает актуальная необходимость проверки корректности применения известных методик и разработки методики синтеза ФНЧ выше 16 порядка.

2. Процедура синтеза согласующего ФНЧ

Пусть требуется для известных r и W рассчитать нормированные параметры g_1, g_2, …, g_{n-1}, g_n, обеспечивающие равноволновую аппроксимацию коэффициента передачи по Чебышёву и требуемую неравномерность коэффициента отражения в заданной рабочей полосе частот. В случае неравных нагрузок ($r\neq1$) характеристика затухания фильтра выглядит, как показано на рис. 4. Такой тип ФНЧ в [6] определён как квазиполосовой ФНЧ.

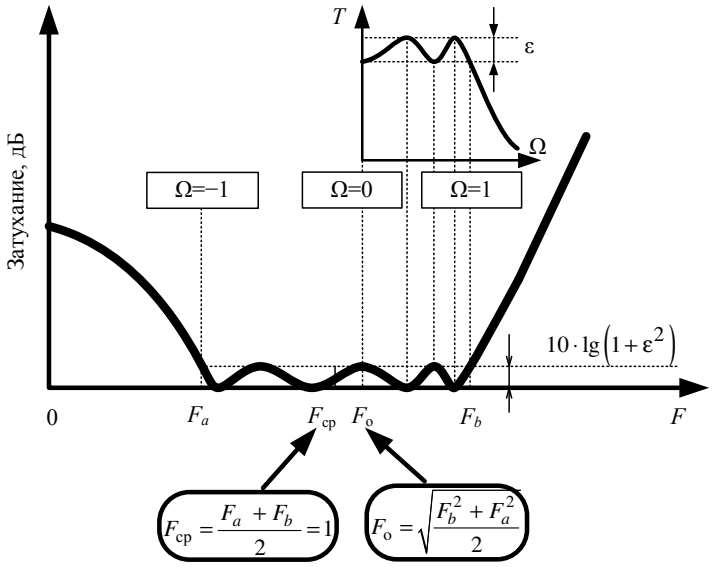


Рис. 4. Формирование характеристики затухания квазиполосового ФНЧ

Основными этапами синтеза согласующего фильтра являются: 1) выбор фильтра прототипа; 2) транспонирование частоты; 3) определение полюсов передаточной функции фильтра прототипа; 4) определение полюсов передаточной функции согласующего фильтра; 5) формирование выражения для коэффициента отражения согласующего фильтра; 6) формирование

выражения для входного (или выходного) сопротивления фильтра; 7) синтез лестничной цепи по методу Кауэра; 8) денормирование значений элементов лестничной цепи.

Передаточная функция ФНЧ прототипа описывается выражением

$$K_p = 1/\left[1 + \varepsilon^2 \cdot T_n^2(\Omega)\right], \tag{1}$$

где ε – коэффициент неравномерности передачи в полосе пропускания, T_n – многочлен Чебышёва первого рода n–го порядка, Ω – нормированная круговая частота.

Для синтеза квазиполосового фильтра следует осуществить транспонирование частоты подстановкой [6]

$$\Omega = \frac{\omega^2 - \omega_o^2}{(\omega_b - \omega_a)\cdot \omega_{ср}} = \frac{\left|\omega^2 - \omega_o^2\right|}{A}, \tag{2}$$

где ω_a – нижняя граница полосы пропускания, ω_b – верхняя граница полосы пропускания, ω – текущая круговая частота, $\omega_o^2 = (\omega_b^2 + \omega_a^2)/2$, $\omega_{ср} = (\omega_a + \omega_b)/2$.

Все значения частот нормируются относительно средней частоты полосы пропускания $\omega_{норм} = \omega_{ср}$, которая принимается равной единице: $\hat{p} = j\hat{\omega} = j(\omega/\omega_{норм})$. В связи с этим можно полагать, что $A = \omega_b - \omega_a$. В [4] и [5] нормирование частот предлагается осуществлять относительно частоты $\omega_{норм} = \omega_o = 1$. Максимальное значение нормированной полосы фильтра прототипа при нормировании относительно $\omega_{ср}$ определяется при $\omega_a = 0$ и равно $\Delta\hat{\omega} = \hat{\omega}_b - \hat{\omega}_a = \hat{\omega}_b = 2$. При нормировании относительно $\omega_o = 1$ максимальное значение нормированной полосы фильтра прототипа равно $\Delta\hat{\omega} = \hat{\omega}_b - \hat{\omega}_a = \hat{\omega}_b = \sqrt{2}$. С учётом выражений для ω_o и $\omega_{ср}$ можно показать, что

$$\Omega = 2 \cdot \frac{\hat{\omega}^2 - \hat{\omega}_o^2}{\hat{\omega}_b^2 - \hat{\omega}_a^2} = 2 \cdot \frac{\hat{\omega}^2 - \hat{\omega}_o^2}{2\hat{\omega}_o^2 - 2\hat{\omega}_a^2} = -\frac{\hat{\omega}_o^2 - \hat{\omega}^2}{\hat{\omega}_o^2 - \hat{\omega}_a^2}$$

$$\text{или } \Omega = 2 \cdot \frac{\hat{\omega}^2 - \hat{\omega}_o^2}{\hat{\omega}_b^2 - \hat{\omega}_a^2} = 2 \cdot \frac{\hat{\omega}^2 - \hat{\omega}_o^2}{2\hat{\omega}_b^2 - 2\hat{\omega}_o^2} = \frac{\hat{\omega}^2 - \hat{\omega}_o^2}{\hat{\omega}_b^2 - \hat{\omega}_o^2}. \tag{3}$$

Из (3) видно, что граничным частотам полосы пропускания $\hat{\omega}_a$ и $\hat{\omega}_b$,

соответствуют значения $\Omega = \pm 1$, а значение $\hat{\omega} = \hat{\omega}_o$ соответствует значению $\Omega = 0$, т.е. начальной (нулевой) частоте фильтра прототипа. Коэффициент передачи на нулевой частоте определяется, исходя из заданных значений сопротивлений источника сигнала $R_с$ и нагрузки $R_н$:

$$K_{po} = 1/\left[1 + \varepsilon^2 \cdot T_n^2(\hat{\omega} = 0)\right] = 1/\left[1 + \varepsilon^2 \cdot T_n^2\left(\Omega = -\frac{\hat{\omega}_b^2 + \hat{\omega}_a^2}{\hat{\omega}_b^2 - \hat{\omega}_a^2}\right)\right] = \frac{4R_с R_н}{(R_с + R_н)^2}, \quad (4)$$

откуда следует выражение для квадрата коэффициента неравномерности передачи прототипа в рабочей полосе:

$$\varepsilon^2 = \left[\frac{(R_с - R_н)^2}{4R_с R_н}\right] \Big/ T_n^2\left(\Omega = -\frac{\hat{\omega}_b^2 + \hat{\omega}_a^2}{\hat{\omega}_b^2 - \hat{\omega}_a^2}\right). \quad (5)$$

Как известно полиномы Чебышёва первого рода степени n в диапазоне значений от $\Omega = -1$ до $\Omega = +1$ определяются рекуррентной формулой $T_n(\Omega) = \cos[n \cdot \arccos(\Omega)]$ и вне полосы через гиперболические функции: $T_n(\Omega) = \mathrm{ch}[n \cdot \mathrm{arch}(\Omega)]$. Так как функция ареа-косинуса не определена для отрицательных значений Ω, появляющихся в (2) при транспонировании частоты в полосах задержания квазиполосового фильтра для $F < F_a$ ($\hat{\omega} < \hat{\omega}_a$) и $F > F_b$ ($\hat{\omega} > \hat{\omega}_b$), то в (2) для этого предусмотрен знак модуля разности $(\omega^2 - \omega_o^2)$.

Характеристика затухания квазиполосового фильтра, т.е. знаменатель квадрата модуля передаточной функции для полосы задержания имеет вид:

$$L = 1 + \varepsilon^2 \mathrm{ch}^2\left[\frac{n}{2} \cdot \mathrm{arch}\left(\frac{\hat{\omega}_o^2 - \hat{\omega}^2}{\hat{\omega}_o^2 - \hat{\omega}_a^2}\right)\right],$$

а для полосы пропускания:

$$L = 1 + \varepsilon^2 \cos^2\left[\frac{n}{2} \cdot \arccos\left(\frac{\hat{\omega}_o^2 - \hat{\omega}^2}{\hat{\omega}_o^2 - \hat{\omega}_a^2}\right)\right].$$

Полюсы передаточной функции фильтра прототипа Чебышёва $\hat{P}_k = \mathrm{Re}(\hat{P}_k) + \mathrm{j}\mathrm{Im}(\hat{P}_k)$ для $k = 1,2,\ldots,n$ определяются из уравнения

$1+\varepsilon^2 T_n^2(\Omega)=0$ с учётом замены $\Omega=(\hat{p}/\mathrm{j})$ [4]:

$$\mathrm{Re}(\hat{P}_k)=\pm\sin\left(\pi\cdot\frac{2k-1}{2n}\right)\cdot\mathrm{sh}\left(\frac{1}{n}\cdot\mathrm{arsh}\left[\frac{1}{\varepsilon}\right]\right);$$

$$\mathrm{Im}(\hat{P}_k)=\cos\left(\pi\cdot\frac{2k-1}{2n}\right)\cdot\mathrm{ch}\left(\frac{1}{n}\cdot\mathrm{arsh}\left[\frac{1}{\varepsilon}\right]\right). \qquad (6)$$

Листинг функций расчёта коэффициента передачи и полюсов фильтра прототипа приведен на рис. 5. Имена функций соответствуют структурной схеме программного модуля на рис. 10.

```
// Расчёт коэффициента передачи фильтра:
function recalK() {
    // Исходные данные: нормированная полоса W, порядок функции n,
    r=R1/R2; // значения сопротивлений Rc=R1 и Rн =R2
    Fa=1-W/2;  Fb=1+W/2; // Расчёт граничных частот
    F0=Math.pow((Fb*Fb+Fa*Fa)/2,0.5); // Центральная частота
    var Faverage=(Fb+Fa)/2; // Средняя частоты НЧ прототипа
    var DDelta=(Fb-Fa); A=DDelta*Faverage; // Расчёт рабочей полосы и параметра A
    // Функция Чебышёва n/2-го порядка на нулевой частоте  a=0
    var TnH0=Math.cosh((n/2)*Math.acosh((F0*F0)/A));
    Ksi=(r-1)/(2*Math.pow(r,0.5))/TnH0; // Неравномерность передачи в рабочей полосе
    for (var Fcur = fmin;Fcur<=fmax;Fcur+=(fmax - fmin) / 800) {var k = funK(Fcur);}
} // 800 - количество точек на графике коэффициента передачи
function funK(a) {
    var aa=Math.abs(F0*F0-a*a)/A; // Трансформирующая частота
    if ((a >= Fa)&&(a <= Fb)){
        var Tn=Math.cos((n/2)*Math.acos(aa)); // функция Чебышёва в рабочей полосе
        Kp=1/(1+Ksi*Ksi*Tn*Tn); // передача в рабочей полосе
    }else{
        var TnH=Math.cosh((n/2)*Math.acosh(aa)); //функция Чебышёва вне полосы
        Kp=1/(1+Ksi*Ksi*TnH*TnH); // передача вне рабочей полосы
    }return Kp;
}
//  Расчёт полюсов прототипа ФНЧ Чебышёва:
function Proots(){
    var tt=n/2; WR = new Array(tt); WI = new Array(tt);
    for (var mm = 1; mm<=tt;mm++){
        var alpha = Math.PI * (2 * mm - 1) / 2 / tt;
        WR[mm] =- Math.sin(alpha) * Math.sinh((1 / tt) * Math.asinh(1 / Ksi));
        WI[mm] = Math.cos(alpha) * Math.cosh((1 / tt) * Math.asinh(1 / Ksi));
    }
 }
```

Рис. 5. Листинг функций расчёта коэффициента передачи recalK, funK и полюсов Proots фильтра прототипа

Для полюсов передаточной функции фильтра прототипа с учётом (2) можно записать:

$$\hat{P}_k = \mathrm{Re}(\hat{P}_k) + j\,\mathrm{Im}(\hat{P}_k) = \Omega_k = -\mathrm{j}\frac{\left(\hat{\sigma}_k + j\hat{\omega}_k\right)^2 + \hat{\omega}_{\mathrm{o}}^2}{A}; \qquad (7)$$

откуда после преобразований для квазиполосового фильтра следует:

$$\left(\hat{\sigma}_k + \mathrm{j}\hat{\omega}_k\right) = \sqrt{-[A\cdot\mathrm{Im}(\hat{P}_k) + \hat{\omega}_{\mathrm{o}}^2] + \mathrm{j}A\cdot\mathrm{Re}(\hat{P}_k)} = \sqrt{x + \mathrm{j}\cdot\mathrm{y}},$$

где $x = -[A\cdot\mathrm{Im}(\hat{P}_k) + \hat{\omega}_{\mathrm{o}}^2];\ y = A\cdot\mathrm{Re}(\hat{P}_k)$.

Используя формулу Муавра, окончательно определяются полюса функции передачи

$$\left(\hat{\sigma}_k + \mathrm{j}\hat{\omega}_k\right) = \sqrt{\rho}\left[\cos\left(\frac{\varphi + 2\pi(k-1)}{2}\right) + \mathrm{j}\sin\left(\frac{\varphi + 2\pi(k-1)}{2}\right)\right], \qquad (8)$$

где

$$\rho = \sqrt{x^2 + y^2} = \sqrt{[A\cdot\mathrm{Im}(\hat{P}_k) + \hat{\omega}_{\mathrm{o}}^2]^2 + [A\cdot\mathrm{Re}(\hat{P}_k)]^2}, \qquad (9)^1$$

$$\varphi = \pi + \mathrm{arctg}\left[\frac{y}{x}\right]. \qquad (10)$$

При расчёте следует учесть неоднозначность определения угла φ:

1) если x > 0 (1–я и 4–я координатные четверти, или правая полуплоскость), то аргумент нужно находить по формуле φ=arctg(y/x);

2) если x < 0, y > 0 (2–я координатная четверть), то аргумент нужно находить по формуле φ = π + arctg(y/x);

3) если x < 0, y < 0 (3–я координатная четверть), то аргумент нужно находить по формуле φ = –π + arctg(y/x).

Таким образом (10) соответствует 2–й координатной четверти.

Выражения, связывающие номинальный коэффициент передачи мощности K_p, коэффициент отражения $\Gamma_{\text{вых}}$ и нормированное выходное сопротивление цепи $\hat{Z}_{\text{вых}} = Z_{\text{вых}} / R_{\text{н}}$, имеют следующий вид:

$$K_p = 1 - \left|\Gamma_{\text{вых}}\right|^2,\ \Gamma_{\text{вых}} = \left(Z_{\text{вых}} - R_{\text{н}}\right)/\left(Z_{\text{вых}} + R_{\text{н}}\right) = \left(\hat{Z}_{\text{вых}} - 1\right)/\left(\hat{Z}_{\text{вых}} + 1\right), \qquad (11)$$

---

1. В [2] на с. 943 в формуле (25) допущена опечатка. Вместо $(A\cdot\sigma''_{bk})^2$ напечатано $A(\sigma''_{bk})^2$

откуда с учётом (1) и (2) следует:

$$\left|\Gamma_{\text{вых}}\right|^2 = 1 - K_p = \left[\varepsilon^2 \cdot T_n^2\left(\Omega = \frac{\left|\hat{\omega}^2 - \hat{\omega}_{\text{o}}^2\right|}{\hat{\omega}_b - \hat{\omega}_a}\right)\right] \Big/ \left[1 + \varepsilon^2 \cdot T_n^2\left(\Omega = \frac{\left|\hat{\omega}^2 - \hat{\omega}_{\text{o}}^2\right|}{\hat{\omega}_b - \hat{\omega}_a}\right)\right]. \quad (12)$$

Нули полинома числителя коэффициента отражения для фильтра прототипа определяются нулями соответствующего многочлена Чебышёва и рассчитываются в соответствии с выражением

$$\hat{Z}_k = \cos\left(\pi \cdot \frac{2k+1}{2n}\right), \quad k = 0,1,...,n-1. \quad (13)$$

Нули будут чисто мнимыми и сопряжёнными $\hat{Z}_k = \pm \mathrm{j}\,\mathrm{Im}\,\hat{Z}_k$, поэтому после применения подстановки (2) нули полинома числителя коэффициента отражения для квазиполосового фильтра определяются из (7) в соответствии с выражением $j\hat{\omega}_{zk} = j\sqrt{A \cdot \mathrm{Im}(\hat{Z}_k) + \hat{\omega}_{\text{o}}^2}$.

Листинг функций расчёта полюсов квазиполосового фильтра и нулей коэффициента отражения фильтра прототипа и квазиполосового фильтра приведен на рис. 6.

Для составления полинома знаменателя коэффициента отражения квазиполосового фильтра из (6) используются полюсы, расположенные в левой полуплоскости (для k = 2). Для упрощения вначале следует сформировать трёхчлены из пар комплексно–сопряжённых полюсов:

$$\begin{gathered}[p - (\mathrm{Re}\,p_k + j\,\mathrm{Im}\,p_k)] \cdot [p - (\mathrm{Re}\,p_k - j\,\mathrm{Im}\,p_k)] = \\ = [(p - \mathrm{Re}\,p_k) - j\,\mathrm{Im}\,p_k] \cdot [(p - \mathrm{Re}\,p_k) + j\,\mathrm{Im}\,p_k] = \\ = (p - \mathrm{Re}\,p_k)^2 + (\mathrm{Im}\,p_k)^2 = p^2 - 2p \cdot \mathrm{Re}\,p_k + \left[(\mathrm{Re}\,p_k)^2 + (\mathrm{Im}\,p_k)^2\right].\end{gathered}$$

Аналогичным образом формируется полином числителя коэффициента отражения с помощью трехчленов:

$$\begin{gathered}[p - (\mathrm{Re}\,z_k + j\,\mathrm{Im}\,z_k)] \cdot [p - (\mathrm{Re}\,z_k - j\,\mathrm{Im}\,z_k)] = \\ = [(p - \mathrm{Re}\,z_k) - j\,\mathrm{Im}\,z_k] \cdot [(p - \mathrm{Re}\,z_k) + j\,\mathrm{Im}\,z_k] = \\ = (p - \mathrm{Re}\,z_k)^2 + (\mathrm{Im}\,z_k)^2 = p^2 - 2p \cdot \mathrm{Re}\,z_k + \left[(\mathrm{Re}\,z_k)^2 + (\mathrm{Im}\,z_k)^2\right].\end{gathered}$$

```
//  Расчёт полюсов квазиполосового ФНЧ:
function Proots2(){
    WR2 = new Array(n);      WI2 = new Array(n);
    for (var mm = 2; mm <= n; mm += 2){
        var ReX =AA*WI[mm/2]+Math.pow(F0,2); var ImX = AA*WR[mm/2];
        var module=Math.pow(Math.pow(ReX,2)+Math.pow(ImX,2),0.5);
        var ugol=Math.atan2(ImX,ReX);
        WR2[mm]=Math.pow(module,1/2)*Math.cos(ugol/2+Math.PI);
        WR2[mm-1]=WR2[mm];
        WI2[mm]=Math.pow(module,1/2)*Math.sin(ugol/2+Math.PI);
        WI2[mm-1]=-WI2[mm];
    }
}
// Расчёт нулей полинома коэффициента отражения прототипа:
function Zroots(){
    var tt=n/2; WZ = new Array(tt);
    for (var mm = 1; mm <= tt; mm ++){WZ[mm]=Math.cos((Math.PI/(2*tt)*(2*mm-1)));}
}
// Расчёт нулей полинома коэффициента отражения ФНЧ:
function Zroots2(){
    WRZ2 = new Array(n); WIZ2 = new Array(n);
    for (var mm = 2; mm<=n;mm+=2){
        WRZ2[mm-1]=0; WRZ2[mm]=0;
        WIZ2[mm-1]=Math.pow(Math.pow(F0,2)+AA*WZ[mm/2],1/2);
        WIZ2[mm]=- Math.pow(Math.pow(F0,2)+AA*WZ[mm/2],1/2);
    }
}
```

Рис. 6. Листинг функций для расчёта полюсов квазиполосового фильтра Proots2, нулей коэффициента отражения фильтра прототипа Zroots и квазиполосового фильтра Zroots2

В результате коэффициент отражения равен:

$$\Gamma_{\text{вых}} = \frac{(\hat{p}-z_1)(\hat{p}-z_2)\cdot\ldots\cdot(\hat{p}-z_n)}{(\hat{p}-p_1)(\hat{p}-p_2)\cdot\ldots\cdot(\hat{p}-p_n)} = \frac{\prod_{k=1}^{n/2}\left[p^2-2p\cdot\operatorname{Im}z_k+(\operatorname{Re}z_k)^2+(\operatorname{Im}z_k)^2\right]}{\prod_{k=1}^{n/2}\left[p^2-2p\cdot\operatorname{Re}p_k+(\operatorname{Re}p_k)^2+(\operatorname{Im}p_k)^2\right]} =$$

$$= \frac{a_n\hat{p}^n+a_{n-1}\hat{p}^{n-1}+\ldots+a_1\hat{p}+a_0}{b_n\hat{p}^n+b_{n-1}\hat{p}^{n-1}+\ldots+b_1\hat{p}+b_0}. \qquad (14)$$

Коэффициенты полиномов определяются по правилам перемножения многочленов в соответствии с выражением $a_k=\sum_{i=0}^{k}a_ib_{k-i};\quad k=0,1,2,\ldots,(l+m);$ где l и m степени перемножаемых полиномов, a и b - их коэффициенты. Листинг функции расчёта коэффициентов полиномов приведен на рис. 7.

```
function coefpol(trinomial){// Расчёт коэффициентов полиномов по их корням
    var Z = new Array(n); var first=[];
    if (nn>2){for (var i=0; i<3; i++){first[i]=trinomial[1][i];}
    }else{Z=trinomial[1];}
    for (var i=1; I <n/2; i++) {
        var second=trinomial[i+1];
        for (var j=0; j <= n; j++){
            Z[j]=0;
            for (var k=0; k<=j; k++){
                if (j-k < second.length){
                    if (k < first.length) {Z[j]=Z[j]+first[k]*second[j-k];}
                }
            }
        }
        first=[]; for (var l = 0; l < Z.length; l++){ first.push(Z[l]);}
    }
    return Z;
}
```

Рис. 7. Листинг функции coefpol расчёта коэффициентов полиномов

Листинг функции формирования числителя и знаменателя коэффициента отражения квазиполосового фильтра приведен на рис. 8.

```
// Формирование знаменателя коэффициента отражения по трёхчленам:
function polynomialP (){
    trinomialP = new Array(n/2); var j=1;
    for(var i=1; I <= (n/2); i++){trinomialP[i]= new Array(3); }
    for (var i = 1; I < trinomialP.length; i++){
        trinomialP[i][0] = WR2[j]*WR2[j]+WI2[j]*WI2[j];
        trinomialP[i][1] = -2*WR2[j]; trinomialP[i][2] = 1; var j=j+2;
    }
    // Расчёт коэффициентов полинома знаменателя коэффициента отражения
    Ppolynomial = new Array(n); Ppolynomial = coefpol(trinomialP);
}
// Формирование числителя коэффициента отражения по трёхчленам:
function polynomialZ(){
    trinomialZ = new Array(n/2);  var j=1;
    for(var i=1; I <= (n/2); i++){  trinomialZ[i]= new Array(3); }
    for (var i = 1; i<trinomialZ.length; i++){
        trinomialZ[i][0] = WRZ2[j]*WRZ2[j]+WIZ2[j]*WIZ2[j];
        trinomialZ[i][1] = -2*WRZ2[j]; trinomialZ[i][2] = 1; var j=j+2;
    }
    // Расчёт коэффициентов полинома числителя коэффициента отражения
    Zpolynomial = new Array(n); Zpolynomial = coefpol(trinomialZ);
}
```

Рис. 8. Листинг функций polynomialZ и polynomialP для формирования числителя и знаменателя коэффициента отражения квазиполосового ФНЧ

В (14) значения старших коэффициентов полиномов равны единице, т.е. $a_n = 1$ и $b_n = 1$. На последнем этапе формируется выражение для выходного сопротивления на основании (11):

$$\hat{Z}_{\text{вых}} = \frac{\left(1 + \dfrac{\hat{p}^n + a_{n-1}\hat{p}^{n-1} + ... + a_1\hat{p} + a_0}{\hat{p}^n + b_{n-1}\hat{p}^{n-1} + ... + b_1\hat{p} + b_0}\right)}{\left(1 - \dfrac{\hat{p}^n + a_{n-1}\hat{p}^{n-1} + ... + a_1\hat{p} + a_0}{\hat{p}^n + b_{n-1}\hat{p}^{n-1} + ... + b_1\hat{p} + b_0}\right)} = \tag{15}$$

$$= \frac{2\hat{p}^n + \hat{p}^{n-1}(b_{n-1} + a_{n-1}) + ... + \hat{p}(b_1 + a_1) + (b_0 + a_0)}{\hat{p}^{n-1}(b_{n-1} - a_{n-1}) + ... + \hat{p}(b_1 - a_1) + (b_0 - a_0)}.$$

Синтез двухполюсников методом Кауэра производится путём представления операторной функции выходного сопротивления $\hat{Z}_{\text{вых}}(\hat{p})$ в виде цепной дроби вида

$$\hat{Z}_{\text{вых}}(\hat{p}) = g_1\hat{p} + \cfrac{1}{g_2\hat{p} + ... + \cfrac{1}{g_{n-1}\hat{p} + \cfrac{1}{g_n\hat{p} + g_{n+1}}}} =$$

$$= \hat{p}\hat{L}_1 + \cfrac{1}{\hat{p}\hat{C}_2 + ... + \cfrac{1}{\hat{p}\hat{L}_{2k-1} + \cfrac{1}{\hat{p}\hat{C}_{2k} + \cfrac{1}{r}}}}, \tag{16}$$

где параметры g_n представляют собой нормированные значения индуктивностей последовательных катушек и ёмкостей параллельных конденсаторов; $\hat{L}_1, \hat{L}_3, ..., \hat{L}_{2k-1}$ и $\hat{C}_2, \hat{C}_4, ..., \hat{C}_{2k}$ – нормированные значения индуктивностей катушек и ёмкостей конденсаторов для $k = 1,...,n/2$:

$$\hat{L}_{2k-1} = g_{2k-1}, \quad \hat{C}_{2k} = g_{2k}. \tag{17}$$

Листинг функции расчёта нормированных параметров квазиполосового ФНЧ приведен на рис. 9.

Полученной дроби соответствует первая каноническая схема Кауэра, изображённая на рис. 1. Окончательный расчёт параметров элементов цепи производится путём денормирования с учётом реальных значений

сопротивления нагрузки $R_н$ и нормирующей частоты фильтра прототипа $f_{норм}$:

$$L_{2k-1} = g_{2k-1} \cdot \frac{R_н}{2\pi f_{норм}};\ C_{2k} = g_{2k} \cdot \frac{1}{2\pi f_{норм} R_н}. \qquad (18)$$

```
function ROutput () {
    //Расчёт коэффициентов полиномов числителя и знаменателя Zвых
    ZRpolynomial = new Array(n); PRpolynomial = new Array(n-1);
    for (var i = n; i>=0; i--){
        ZRpolynomial[i]=Ppolynomial[i] + Zpolynomial[i]; // числитель Zвых
        // Разность полиномов: это знаменатель Zвых
        if (i-1>=0) {PRpolynomial[i-1]=Ppolynomial[i-1] -Zpolynomial[i-1];}
    }
    // Разложение в цепную дробь
    G = new Array(n+1); / /матрица нормированных параметров
    var i=1; var nnn=n; var divider=PRpolynomial;
    var remainder=ZRpolynomial; var difference=[]; var Pdivider=[];
    while (nnn>=0) {
        G[i]=Number((remainder[remainder.length-1]/divider[divider.length-1]).toFixed(18));
        if (i==nn+1){G[i]=r;}
        for (var j=divider.length-1; j>=0; j--){
            Pdivider[j]=Number((divider[j]*G[i]).toFixed(18));
        }
        Pdivider.unshift(0);
        for (var j=remainder.length-1; j>=0; j--){
            difference[j]=Number((remainder[j]-Pdivider[j]).toFixed(18));
        }
        difference.pop(); if (divider.length!=1) {difference.pop();}
        var remainder=[]; remainder=divider; var divider=[];
        var divider=difference; nnn--; i++; var Pdivider=[]; var difference=[];
    }
}
```

Рис. 9. Листинг функции ROutput для расчёта нормированных параметров квазиполосового ФНЧ

Формулы (1)–(18) представляют собой математическую модель для решения задачи синтеза ФНЧ с помощью компьютерной программы. Структурная схема программного модуля представлена на рис. 10.

3. Анализ результатов расчёта с помощью стандартной математики

Разложение функции входного сопротивления в виде непрерывной дроби приводит к постепенной потере точности. Основная причина потери точности заключается в том, что при использовании чисел с плавающей запятой в формате *double* длина мантиссы составляет 52 двоичных разряда, что соответствует 16–ти верным знакам мантиссы после десятичной запятой.

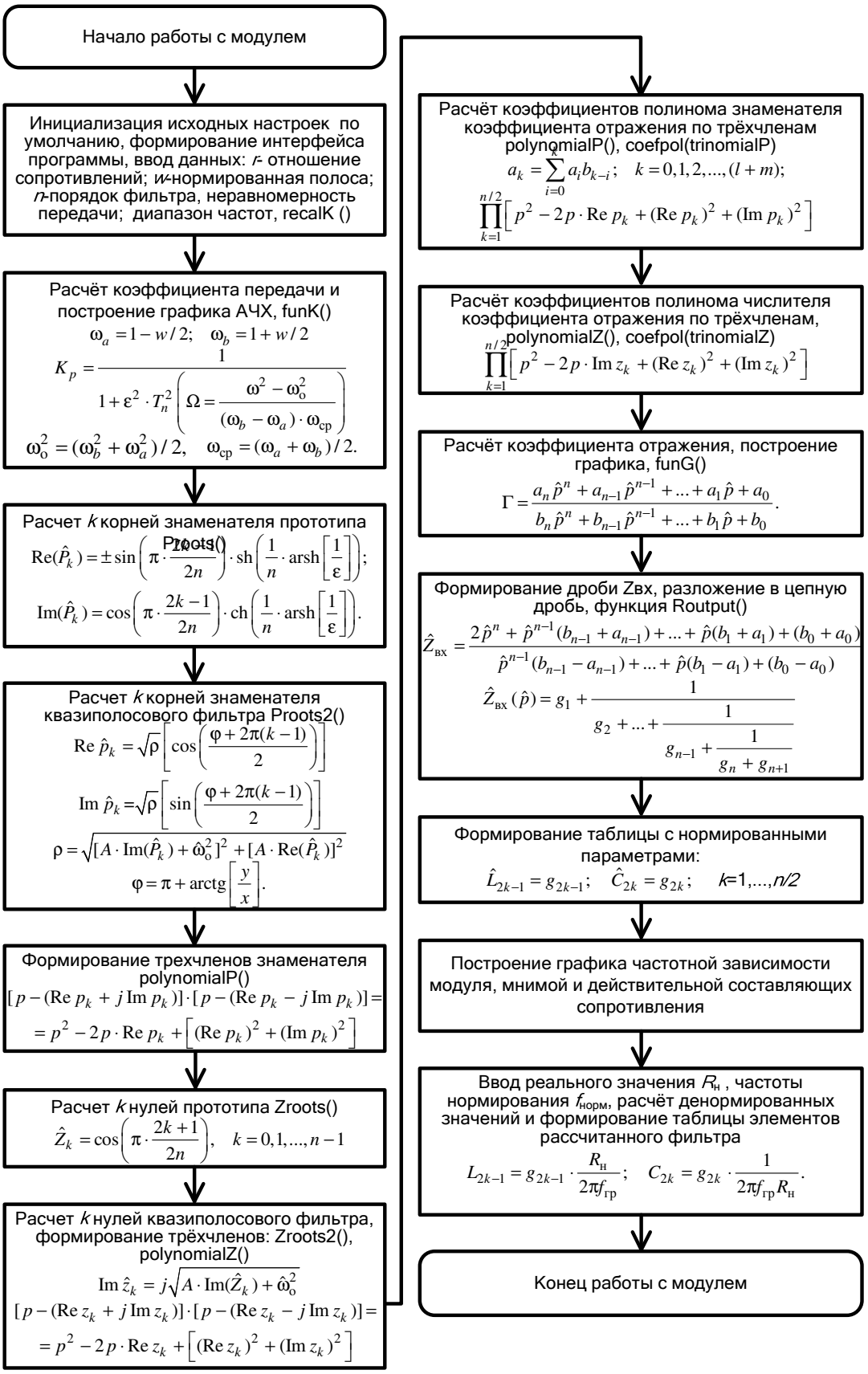


Рис. 10. Структурная схема программного модуля

Наибольшие погрешности вызывает вычитание соизмеримых чисел. Последний знак мантиссы не является точным. Значение для g_1 получается с большей точностью, а все последующие значения g_n получаются с меньшей точностью.

Разработана JavaScript программа, которая позволила проверить табулированные в [2] данные и установить, что данным для фильтров 8…10–го порядков не следует доверять уже с первого–второго знака после десятичной запятой! На рис. 6. представлена АЧХ фильтра 10–го порядка для r=50 и w=0.3, полученная в программе Filter Solutions 8.0 [1] по данным уже разработанной программы. Полученный график подтверждает корректность разработанной модели. Недостатки фильтра, рассчитанного по данным [2], уже отмечались выше (см. рис.3).

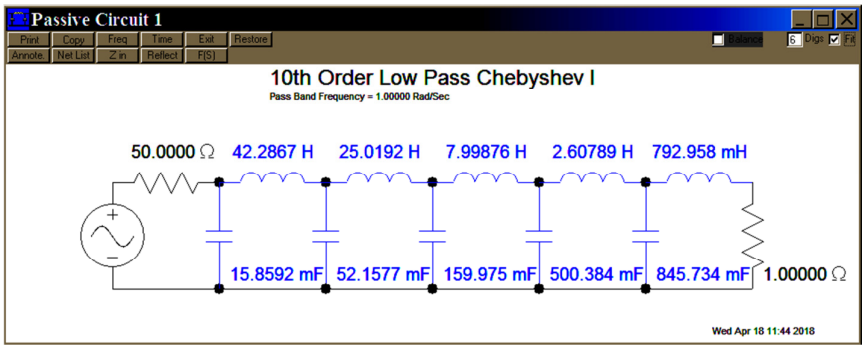
В таблице 2 представлены результаты расчёта параметров g_n фильтра 20–го порядка для w=0,3 и r = 5 в зависимости от числа знаков в мантиссе переменной с плавающей запятой.

Ограничение числа знаков после десятичной запятой осуществлено программным способом. Достоверными считаются знаки, повторяющиеся во всех мантиссах различной длины. Для m = 10…16 знаки, заслуживающие доверия, выделены жирным шрифтом красного цвета (если в качестве эталонных цифр взять результат из левой колонки для m=18). Как видно из табл. 2 одна итерация увеличивает число ненадёжных знаков примерно на одну единицу. В результате для 16 знаков мантиссы (стандартная компьютерная математика) синтез структуры с n = 20 оказывается невозможным. Левая колонка табл. 2 (для m = 18), начиная с g_{16} и далее, скорее всего, тоже содержит большое количество неверных знаков. Свойство антиметричности лестничной структуры, при котором для нечётных элементов выполняется соотношение $g_{n+1-k} = g_k / r$, а для чётных – $g_{n+1-k} = g_k r$, позволяет оценить точность разложения входного сопротивления в цепную дробь. Из таблицы 2 для n = 18 следует, что

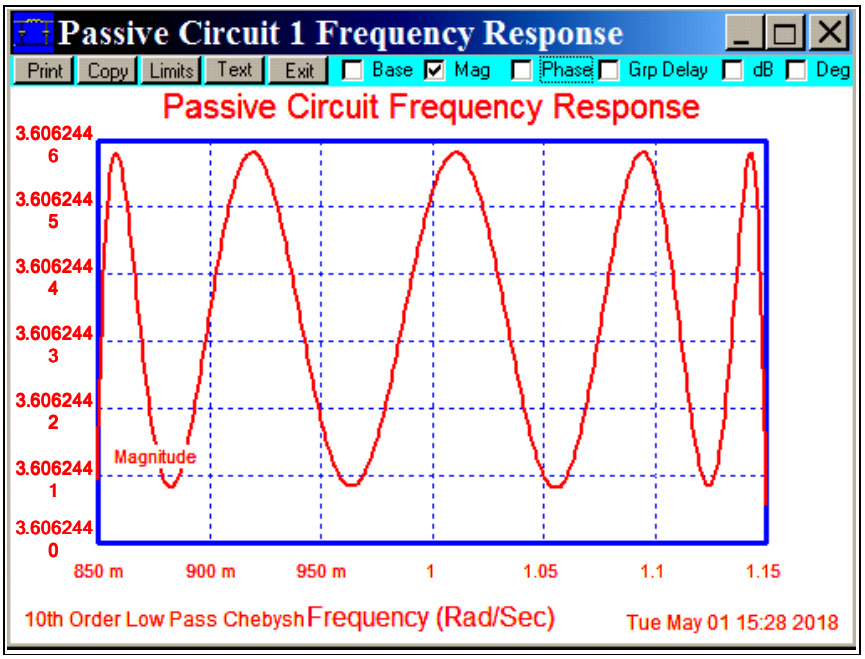
$$g_{20} = g_1/5 = 0{,}2954195543744352/5 = 0{,}05908391087488704,$$

$$g_{19} = g_2 \cdot 5 = 0{,}713461109564948 \cdot 5 = 3{,}56730554782474,$$

что не совпадает с полученными в результате разложения значениями g_{20} = 0,032916230753412865 и g_{19} = 5.54123352930229.



а



б

Рис. 6. АЧХ фильтра 10–го порядка

О потере точности расчётов говорит и появление в процессе разложения коэффициентов с отрицательными значениями (например: g_{13}, g_{14} для m = 10; g_{17}, g_{18} для m = 14).

Таблица 2. Значения нормированных параметров лестничной структуры n=20

g_n	Длина мантиссы m, знаков				
	18	**16**	**14**	**12**	**10**
g_1	**0.2954195543744352**	**0.295419554374**4352	**0.2954195543744**4	**0.295419554374**	**0.295419554**4
g_2	**0.713461109564948**	**0.713461109564**9473	**0.71346110956**499	**0.71346110**9561	**0.713461109**8
g_3	**1.0058241561838035**	**1.005824156183**7966	**1.00582415618**413	**1.0058241**56152	**1.005824158**1
g_4	**1.0778527135007734**	**1.077852713500**736	**1.07785271350**291	**1.07785271**3286	**1.077852726**6
g5	**1.3444551271939031**	**1.34445512719**36078	**1.3444551272**1384	**1.3444551**25133	**1.344455**2542
g_6	**1.1116842558930766**	**1.1116842558**91377	**1.1116842560**1542	**1.111684**43014	**1.111685**0547
g_7	**1.7143743392518236**	**1.714374339**2240474	**1.714374**34115213	**1.71437**4137835	**1.7143**868615
g_8	**0.9418886559187652**	**0.941888655**7877248	**0.941888**66449494	**0.94188**7744902	**0.941**9453415
g_9	**2.3632836433824114**	**2.363283**639323256	**2.36328**390591041	**2.3632**55746937	**2.36**50210613
g_{10}	**0.686285986518519**	**0.686285**9767247607	**0.6862**8661828577	**0.6862**18857231	**0.6**904969103
g_{11}	**3.43143008776028**	**3.4314**29449037259	**3.4314**7128394913	**3.4**27058873063	**3**.7323116007
g_{12}	**0.4726569204179655**	**0.47265**61909475649	**0.472**70397320746	**0.4**67724626061	2.4216582208
g_{13}	**4.709466618031227**	**4.709**377926992177	**4.7**1519519287878	**4**.183521806709	–0.0157276474
g_{14}	**0.342888872823455**	**0.3428**356584472274	**0.34**636667160111	0.19113105171	–1.8857040979
g_{15}	**5.560720624648062**	**5.5**519951966932135	6.20632346322519	1.869831540234	5.4401419765
g_{16}	**0.269723387652699**	**0.26**6596319236514	92.8313518646038	0.219631316503	0.3362139249
g_{17}	**5.5117714667155475**	**5**.082558302126313	–0.00000492164538	5.527060239925	5.8435977804
g_{18}	**0.226485949610610 03**	0.1610284249850325	–92.50550539104933	0.270098171961	0.272047304
g_{19}	**5.54123352930229**	2.776428788285468	6.44373312115062	5.803602803799	5.8088384512
g_{20}	**0.032916230753412865**	0.1015547775683366	0.19975360242714	0.196154248425	0.196187588

Таким образом, синтез лестничных структур высокого порядка методами Кауэра ограничен спецификой компьютерного представления данных. Преодолеть эти ограничения можно, используя программно–аппаратное обеспечение, позволяющее реализовать математические операции над переменными с произвольной длиной мантиссы – математику произвольной или многократной точности (multiple precision). Следует отметить, что

ситуации, требующие реализацию широкополосных согласующих фильтров именно высокого порядка при значительном различии согласуемых сопротивлений нередки. Порядок трансформирующего согласующего фильтра определяется в соответствии с формулой

$$n = 2\,\mathrm{Arch}\left(\frac{r-1}{2\sqrt{r(10^{0,1\alpha}-1)}}\right) \Big/ \mathrm{Arch}\left(\frac{w^2+1}{w^2-1}\right), \qquad (19)$$

где r – отношение согласуемых сопротивлений; α – неравномерность передачи в дБ; w – отношение граничных частот рабочей полосы.

В соответствии с (19) для r=5 и r=50 при α = 1 дБ; w=19 порядок фильтра должен превышать значения 24 и 50, соответственно. Синтезировать фильтры такого порядка средствами стандартной математики по методам Кауэра не представляется возможным.

Числа произвольной длины, например, поддерживаются в библиотеках *GNU Multiple Precision Arithmetic Library* (*GMP*) [8], *BCMath* (*Mathematical Binary Calculator*) [9] и *BigNumber* [10], где числа чаще всего представляются в виде переменных строкового типа. Над ними осуществляются арифметические операции по школьным правилам “в столбик”. По завершении расчётов производится обратное преобразование строк в обычные числа. Плата за применение математики произвольной точности – рост времени расчёта.

Существуют версии *BCMath* для языка *JavaScript* и для языка *PHP*, причём библиотека *BCMath* встроена в *PHP*, начиная с версии 4.0.4. Библиотека *BigNumber* написана на языке C++, но существует её версия для языка *JavaScript*. Версия *BCMath* для *JavaScript* на момент написания статьи продемонстрировала наличие досадных логических ошибок в программном коде функции деления *bcdiv*(), хотя аналогичная версия для *PHP* 5.0 выполняла все расчёты безошибочно. Было решено остановиться на библиотеке *BigNumber* для *JavaScript*. Следует отметить, что указанные выше библиотеки не предусматривают расчёты тригонометрических, гиперболических и логарифмических функций. В разработанной программе реализации этих

функций основаны на вычислении рядов Маклорена, включая расчёт таких констант как π, e, ln10, ln2 и т.д. Листинги функций расчёта фильтра прототипа, квазиполосового ФНЧ, тригонометрических и гиперболических функций с переменными BigNumber для математики произвольной точности приведены на рис. 11- 23.

```
function funK(a) {// расчёт коэффициента передачи
    BigNumber.config({ DECIMAL_PLACES: nn*2 });// длина переменных BigNumber
    e=BigNumber('1e-150'); // точность в рядах Маклорена
    Ln10= LnConstX(10); Ln2= LnConstX(2); // расчёт констант
    pi0=((AtanBig(BigNumber('1').div('2'))).plus(AtanBig(BigNumber('1').div('3')))).times('4');
    var aBig=BigNumber(a.toString());Fa=1-W/2; Fb=1+W/2;
    FaBig=BigNumber('1').minus(BigNumber(W).div('2'));// рабочая полоса
    FbBig=BigNumber('1').plus(BigNumber(W).div('2'));
    //Квадрат центральной частоты квазиполосовой характеристики
    F02Big=((FaBig.pow(2)).plus(FbBig.pow(2))).div('2');
    ABig=DeltaBig.times(FaverageBig); var ttBig=BigNumber(nn/2.toString);
    var aaBig=((F02Big.minus(aBig.pow('2'))).abs()).div(ABig);//Трансформирующая частота
    if ((a >= Fa)&&(a <= Fb)){ // передача в рабочей полосе
        var Tn=(CosBig(ttBig.mul(ACosBig(aaBig)))).toDigits(nn+5);
        Kp=(((KsiBig.mul(KsiBig)).mul(Tn.mul(Tn))).plus('1')).pow(-1);
    }else{// передача вне рабочей полосы
        var TnH=(CoshBig(ttBig.mul(ACoshBig(aaBig)))).toDigits(nn+5);
        Kp=(((KsiBig.mul(KsiBig)).mul(TnH.mul(TnH))).plus('1')).pow(-1);}
    return Kp;
}
function Proots(){// расчёт полюсов прототипа
    var tt=nn/2; var InvttBig=BigNumber('1').div(BigNumber(tt.toString()));
    var InvKsiBig=BigNumber('1').div(KsiBig);WR = new Array(tt);WI = new Array(tt);
    for (var mm = 1; mm<=tt;mm++){
        var temp1=BigNumber(nn.toString());
        var temp2=(((BigNumber(mm.toString())).times('2')).minus('1')).div(temp1);
        var alpha =temp2.times(pi0); var Gamma=(InvttBig.times(ASinhBig(InvKsiBig)));
        WR[mm]=SinBig(alpha).times((SinhBig(Gamma)).neg());
        WI[mm]=CosBig(alpha).times(CoshBig(Gamma));
    }
}
```

Рис. 11. Листинги функций recalK, funK, Proots с переменными BigNumber

```
function  ASinhBig(x) {// расчёт функции гиперболического арксинуса
    var xBig=BigNumber(x.toString());
    var temp=LnBig2(xBig.plus(((xBig.pow(2)).plus('1')).sqrt()));
    return temp.toDigits(stepen);
}
function  ACoshBig(x) {// расчёт функции гиперболического арккосинуса
    var xBig=BigNumber(x.toString());
    var temp=LnBig2(xBig.plus(((xBig.pow(2)).minus('1')).sqrt()));
    return temp.toDigits(stepen);
}
```

Рис. 12. Листинг функций гиперболических арксинуса и арккосинуса

```
function Proots2(){// расчёт полюсов квазиполосового фильтра
    WR2 = new Array(nn);    WI2 = new Array(nn); var ABig=BigNumber(A.toString());
    for (var mm = 2; mm<=nn;mm+=2){
        var ReX =BigNumber('-1').times((ABigNumber.times(WI[mm/2])).plus(F02Big));
        var ImX = ABig.times(WR[mm/2]);
        var module=(ReX.pow(2).plus(ImX.pow(2))).sqrt();
        var ugol=Atan2Big(ImX,ReX);
        WR2[mm]=(module.sqrt()).times(CosBig((ugol.div('2')).plus(pi0)));
        WI2[mm]=(module.sqrt()).times(SinBig((ugol.div('2')).plus(pi0)));
        WR2[mm-1]=WR2[mm];WI2[mm-1]=(WI2[mm]).neg();
    }
    // Формирование трехчленов по парам комплексно-сопряжённых корней
    //число трёхчленов вида "A(2)p^2+A(1)p+A(0)" равно nn/2, т.е половине числа
    //комплексно-сопряжённых корней: A(2)=1, A(1)=-2*Re(p1), A(0)=Re(p1)^2+Im(p1)^2,
    trinomialP = new Array(tt); var j=1;
    for(var i=1; i<=tt; i++){trinomialP[i]= new Array(3); }
    for (var i = 1; i<trinomialP.length; i++){
        trinomialP[i][0] = ((WR2[j]).pow('2')).plus(WI2[j].pow('2'));
        trinomialP[i][1] = (WR2[j]).times(BigNumber('-2'));
        trinomialP[i][2] = BigNumber('1'); var j=j+2;
    }
}
function Zroots(){ // расчёт нулей прототипа
    WZ = new Array(tt);
    for (var mm = 1; mm<=tt;mm++){
        WZ[mm]=CosBig(BigNumber(2*mm-1).times(pi0.div(BigNumber(nn))));
    }
}
```

Рис. 13. Листинги функций Proots2, Zroots с переменными BigNumber

```
function coefpol(trinomial){ // расчёт коэффициентов полиномов по трёхчленам
    var Z = new Array(nn); var first=[]; var second=[]; var temp=[];
    if (nn>2){
        for (var i=0; i<3; i++){
            temp[i]=(trinomial[1][i]).toString(); first[i]=BigNumber(temp[i]);}
    }else{Z=BigNumber(trinomial[1]); return Z;  }
    for (var i=1; i<tt; i++) {
        for (var j=0; j<3; j++){
            temp[j]=(trinomial[i+1][j]).toString();second[j]=BigNumber(temp[j])}
        for (var j=0; j<=nn; j++)  {
            Z[j]=BigNumber('0');
            for (var k=0; k<=j; k++){
                if (j-k<second.length) {
                    if (k<first.length) {
                        Z[j]=(Z[j]).plus((first[k]).times(second[j-k]));
                        Z[j]=Z[j].toDigits(stepen);}
                }
            }
        }
        first=[]; for (var l = 0; l < Z.length; l++){first.push(Z[l]);
        }
    }
    return Z;
}
```

Рис. 14. Листинг функции coefpol() с переменными BigNumber

```
function  polinomialG(){ //Расчёт коэффициентов полинома коэффициента отражения
    Ppolinomial = new Array(nn);Ppolinomial = coefpol(trinomialP); //знаменатель
    Zpolinomial = new Array(nn);Zpolinomial = coefpol(trinomialZ); // числитель
    //Расчет коэффициентов полинома входного сопротивления
    ZRpolinomial = new Array(nn); PRpolinomial = new Array(nn-1);
    for (var i = nn; i>=0; i--){
        // Сумма полиномов: это числитель
        ZRpolinomial[i]=(BigNumber(Ppolinomial[i])).plus(BigNumber(Zpolinomial[i]));
        // Разность полиномов: это знаменатель
        var temp=BigNumber(Zpolinomial[i-1]);
        if (i-1>=0) {PRpolinomial[i-1]=(BigNumber(Ppolinomial[i-1])).minus(temp);
        }
    }
    G = new Array(nn+1); // матрица нормированных gn
    var i=1; var nnn=nn; var divider=PRpolinomial;
    var remainder=ZRpolinomial; var difference=[];
    var Pdivider=[];
    while (nnn>=0) {// разложение в цепную дробь
        var temp1= BigNumber(remainder[remainder.length-1]);
        var temp2= BigNumber(divider[divider.length-1]);
        G[i]= temp1.dividedBy(temp2);
        if (i==nn+1){ G[i]=BigNumber(r);}
        for (var j=divider.length-1; j>=0; j--){
            var temp3=BigNumber(divider[j]);
            Pdivider[j]=(temp3.times(G[i])).toDigits(stepen);}
        Pdivider.unshift(0);
        for (var j=remainder.length-1; j>=0; j--){
            var temp4=BigNumber(remainder[j]);
            difference[j]=(temp4.minus(BigNumber(Pdivider[j]))).toDigits(stepen);}
        difference.pop();
        if (divider.length!=1) {difference.pop();}
        var remainder=[];   remainder=divider; var divider=[];
        var divider=difference; nnn--;
        i++;  Pdivider=[];  difference=[];
    }
}
```

Рис. 15. Листинг функции polinomialG с переменными BigNumber

```
function  SinBig(x) {// расчёт функции синуса, x - угол в радианах, e - точность
    var  i=1; var SinBig=BigNumber(x.toString());   var stx=BigNumber(x.toString());
    var stx2=stx;
    if ((stx2.toString()).substr(0,1)=="-") {
        stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
        stx2=BigNumber(stx2);}
    while (stx2.comparedTo(e)==1) {
        i++; var temp=BigNumber(2*i-1).times(2*i-2);
            stx=(stx.neg()).times(BigNumber(x).pow(2)).dividedBy(temp);
        stx=stx.toDigits(stepen);SinBig=SinBig.plus(stx);stx2=stx;
        if ((stx2.toString()).charAt(0)=="-") {
            stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
            stx2=BigNumber(stx2);
        }
    }
    return SinBig;
}
```

Рис. 16. Листинг функции синуса

```
function  CosBig(x) {// расчёт функции косинуса, x - угол в радианах
    var  i=0; var CosBig= BigNumber('1'); var stx=BigNumber('1');
    var stx2=stx; var xBig=BigNumber(x.toString())
    while (stx2.comparedTo(e)==1) { // e - точность
        i++; var temp=BigNumber(2*i).times(2*i-1);
        stx=((stx.neg()).times(xBig.pow(2)).dividedBy(temp).toDigits(stepen);
        CosBig=CosBig.plus(stx); stx2=stx;
        if ((stx2.toString()).charAt(0)=="-") {
            stx2= BigNumber( (stx2.toString()).substring(1,(stx2.toString()).length));
        }
    }
    return CosBig;
}
```

Рис. 17. Листинг функции косинуса

```
function  ASinBig(x) {// расчёт функции арксинуса, x  - угол в радианах
    var  i=0; var xBig=BigNumber(x.toString());
    var  temp1=(BigNumber('1')).minus(xBig.mul(xBig)).sqrt();
    var temp=((BigNumber('1').minus(temp1)).div(BigNumber('2'))).sqrt();
    var ASinBig=temp; var stx=temp; var stx2=stx;
    if ((stx2.toString()).charAt(0)=="-") {
        stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
        stx2=BigNumber(stx2);
    }
    while (stx2.comparedTo(e)==1) {
        i++; var temp2= stx.times(temp.pow(2));
        var temp3=temp2.times((2*i-1)*(2*i-1)).toString();
        stx=(temp3.div((2*i)*(2*i+1)).toString()).toDigits(stepen); stx2=stx;
        if ((stx2.toString()).charAt(0)=="-") {
            stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
            stx2=BigNumber(stx2);
        }
    }
    return ASinBig.mul(BigNumber('2'));
}
function  ACosBig(x) {// расчёт функции арккосинуса
    var temp=pi0.div('2');
    return temp.minus(ASinBig(x));
}
```

Рис. 18. Листинг функций арксинуса и арккосинуса

```
function  CoshBig(x) {// расчёт функции гиперболического косинуса
    var  i=0; var xBig=BigNumber(x.toString()); var CoshBig=BigNumber('1');
    var stx=BigNumber('1'); var stx2=stx;
    while (stx2.comparedTo(e)==1) {
        i++; var temp1= BigNumber(((2*i)*(2*i-1)).toString());
        stx=((stx.times(xBig.pow(2))).div(temp1).toDigits(stepen);
        CoshBig=CoshBig.plus(stx); stx2=stx;
        if ((stx2.toString()).charAt(0)=="-") {
            stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
            stx2=BigNumber(stx2);
        }
    }
    return CoshBig.toDigits(stepen);
}
```

Рис. 19. Листинг функции гиперболического косинуса

```
function  Atan2Big(y,x) {
    var i=0; var xBig=BigNumber(x.toString()); var yBig=BigNumber(y.toString());
    if (((yBig.abs()).div(xBig.abs())).gt(1)) {var xyBig=(xBig.div(yBig)).abs();
    }else{var xyBig=(yBig.div(xBig)).abs();}
    var Atan2Big=xyBig.div((xyBig.pow('2')).plus('1')); var stx=Atan2Big; var stx2=stx;
    while (stx2.comparedTo(e)==1) {
        i++; var temp1=(BigNumber((2*i).toString()));
        var temp2=xyBig.pow('2').div(xyBig.pow('2').plus('1'));
        var temp3=temp2.times(temp1.div(BigNumber((2*i+1).toString())));
        stx=(stx.times(temp3)).toDigits(stepen);stx2=stx;
        Atan2Big=Atan2Big.plus(stx);
        if ((stx2.toString()).charAt(0)=="-") {
            stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
            stx2=BigNumber(stx2);
        }
    }
    if (((yBig.abs()).div(xBig.abs())).gt(1)){Atan2Big=(pi0.div('2')).minus(Atan2Big);}
    // проверка квадранта: если ReX>0 & ImX>0, то угол от 0 до 90;
    // если ReX<0 & ImX>0, то угол от 90 до 180; если ReX<0 & ImX<0, то угол от 180 до
    // 270; если ReX>0 & ImX<0, то угол от 270 до 0
    if ( xBig.gt(0) &&  yBig.gt(0)){Atan2Big=Atan2Big;}
    if (xBig.isNeg() && yBig.gt(0)){Atan2Big=pi0.minus(Atan2Big);}
    if (xBig.isNeg() && yBig.isNeg()){Atan2Big=Atan2Big.minus(pi0);}
    if (xBig.gt(0) && yBig.isNeg()){Atan2Big=Atan2Big.neg();}
    return Atan2Big.toDigits(stepen);
}
```

Рис. 20. Листинг функции арктангенса

```
function  SinhBig(x) {// расчёт функции гиперболического синуса
    var  i=0; var xBig=BigNumber(x.toString()); var SinhBig=xBig;
    var stx=xBig; var stx2=stx;
    if ((stx2.toString()).substr(0,1)==="-") {
        stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
        stx2=BigNumber(stx2);    }
    while (stx2.comparedTo(e)==1) {
        i++; var temp1=(2*i*(2*i+1)).toString(); var temp2=xBig.pow(2);
        stx=stx.times(temp2.div(temp1));stx=stx.toDigits(stepen);
        SinhBig=SinhBig.plus(stx);stx2=stx;
        if ((stx2.toString()).charAt(0)=="-") {
            stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
            stx2=BigNumber(stx2);
        }
    }
    return SinhBig.toDigits(stepen);
}
```

Рис. 21. Листинг функций гиперболического синуса

```
function  LnBig(x) {// расчёт функции натурального логарифма
    var  i=0; var xBig=BigNumber(x.toString());
    if (xBig.lt(1)) {var xBigI=xBig.pow(-1) }else{var xBigI=xBig}
    if (x.isBigNumber){
        var Delit=((Math.trunc(xBigI.toNumber()).toString()).length-1).toString();
    }else { var Delit=((Math.trunc(xBigI).toString()).length-1).toString();}
    // сдвигаем десятичные разряды на Delit позиций влево: x=x/(10^Delit)
    var AxBig=xBigI.div(BigNumber('10').pow(Delit)); var temp= AxBig.toString();
    // продолжаем уменьшать число, деля на 2 в степени Delit2: x=x/(2^Delit2)
    var Delit2=0;
    while (AxBig.gt(1)){AxBig=AxBig.div(2); Delit2++;}
    AxBig=(AxBig.minus('1')).div(AxBig.plus('1'));
    var LnBig=AxBig; var stx=AxBig; var stx2=stx;
    while ((stx2.abs()).comparedTo(e)==1) {
        i++; var temp1=BigNumber((2*i-1).toString());
        var temp2=temp1.div(BigNumber((2*i+1).toString()));
        stx=(stx.times(AxBig.pow('2')).times(temp2)).toDigits(stepen);
        LnBig=LnBig.plus(stx); stx2=stx;
        if ((stx2.toString()).charAt(0)=="-") {
            stx2=(stx2.toString()).substring(1,(stx2.toString()).length);
            stx2=BigNumber(stx2);
        }
    }
    var temp3= BigNumber(Delit).times(Ln10);
    LnBig=(LnBig.times(2)).plus((BigNumber(Delit2).times(Ln2)).plus(temp3));
    if (xBig.lt(1)) {   LnBig=LnBig.neg();}
    return LnBig.toDigits(stepen);
}
```

Рис. 22. Листинг функции натурального логарифма

```
function funG(a) {
    var aBig=BgNumber(a.toString(10));
    // четная часть полинома знаменателя Г:
    var LGZ=BgNumber('0');
    var znak=BgNumber('1');
    for (var i=0;i<=nn;i+=2){
        LGZ=LGZ.plus(Ppolinom[i].mul((aBig.pow(i)).mul(znak)));
        znak=znak.neg();
    }
    var znak=BgNumber('1');
    // нечетная часть полинома знаменателя Г:
    var RGZ=BgNumber('0');
    for (var i=1;i<=nn-1;i+=2){
        RGZ=RGZ.plus(Ppolinom[i].mul((aBig.pow(i)).mul(znak)));
        znak=znak.neg();
    }
    var Znamenatel=((LGZ.pow(2)).plus(RGZ.pow(2))).sqrt();
    // четная часть числителя знаменателя Г:
    var LG=BgNumber('0');
    var znak=BgNumber('1');
    for (var i=0;i<=nn;i+=2){
        LG=LG.plus(Zpolinom[i].mul((aBig.pow(i)).mul(znak)));
        znak=znak.neg();
    }
    var znak=BgNumber('1');
    // нечетная часть числителя знаменателя Г:
    var RG=BgNumber('0');
    for (var i=1;i<=nn-1;i+=2){
        RG=RG.plus(Zpolinom[i].mul((aBig.pow(i)).mul(znak)));
        znak=znak.neg();
    }
    var Chislitel=((LG.pow(2)).plus(RG.pow(2))).sqrt();
    return Chislitel.div(Znamenatel);
}
```

Рис. 23. Листинг функции funG

4. Анализ результатов расчёта для математики произвольной точности

В таблице 3 представлены результаты расчёта значений g_n в зависимости от числа знаков в мантиссе для отношения согласуемых сопротивлений $r = 5$ в нормированной полосе 0,3.

Таблица 3. **Значения нормированных параметров лестничной структуры n=20**

g_n	Число знаков мантиссы, m		
	35	25	20
g_1	**0.29541955437443521234270754113370597**	**0.2954195543744352124083667**	**0.295419554374419181 25**
g_2	**0.71346110956494770514221366391859639**	**0.7134611095649477052322662**	**0.7134611095649 2562165**
g_3	**1.00582415618379906860783798845944071**	**1.0058241561837990687468833**	**1.005824156183765120 04**
g_4	**1.07785271350076178568663714200750203**	**1.0778527135007617857002412**	**1.077852713500 75846525**
g_5	**1.34445512719388538671280141702242219**	**1.3444551271938853869693529**	**1.34445512719 38227563**
g_6	**1.11168425589288948820596037242747494**	**1.1116842558928894880257848**	**1.1116842558 93352591**
g_7	**1.71437433924616586808365116126544162**	**1.7143743392461658687912393**	**1.714374339 24599381954**
g_8	**0.94188865588618165813373786619549812**	**0.9418886558861816576579313**	**0.941888655886 30104095**
g_9	**2.36328364232728559753138888157461324**	**2.3632836423272855992086425**	**2.363283642 32697440206**
g_{10}	**0.68628598394432447400787491979955783**	**0.6862859839443244726151504**	**0.68628598394 490111828**
g_{11}	**3.43142991972162246337327736981691306**	**3.4314299197216224214887443**	**3.43142991 974727706806**
g_{12}	**0.47265672846545710665015125692951756**	**0.4726567284654570527166397**	**0.472656728 49624487801**
g_{13}	**4.709443279430908418764184740187739 42**	**4.7094432794309019577735978**	**4.70944328315065456257**
g_{14}	**0.342874867849233164290631861250067 68**	**0.3428748678492292814490132**	**0.34287487008270803596**
g_{15}	**5.55842127946444759221709261627255 45**	**5.5584212794638105226958762**	**5.5584216459577373771 3**
g_{16}	**0.26889102543877707002874336014120063**	**0.2688910254385472780322545**	**0.2688911576319167343 2**
g_{17}	**5.389263567503809075011186357662297 38**	**5.3892635674708735888869745**	**5.389282514511685971 75**
g_{18}	**0.20116483123675980824844619535676139**	**0.2011648312309020996803515**	**0.2011682011105988173 6**
g_{19}	**3.56730554782473862267980868539358304**	**3.5673055475861367207881386**	**3.56744282237718140891**
g_{20}	**0.05908391087488704**086303198313051947	**0.0590839108**809784763816973	**0.05908**04065438504839

Достоверные знаки выделены жирным шрифтом и повторяются во всех

мантиссах различной длины. Значения нормированных параметров для $m = 35$ сравнивались с результатами для $m = 45$. Согласно свойству антиметричности лестничной структуры

$$g_{20} = g_1/5 = = 0{,}29541955437443521234270754113370597/5 =$$
$$= 0{,}059083910874887024685415082266,$$
$$g_{19} = g_2 \cdot 5 = 0{,}71346110956494770514221366391859639 \cdot 5 =$$
$$= 3{,}567305547824738525711068319 59,$$

что совпадает с полученными в результате разложения данными таблице 3 до 15−го знака.

С точки зрения благоразумной достаточности экспериментальным путём установлено, что для получения не менее 15 достоверных знаков после десятичной запятой для параметров фильтра g_n необходимо увеличить длину мантиссы переменных до значения $2n$.

5. Заключение

Рассмотрены особенности синтеза лестничных трансформирующих ФНЧ высокого порядка по методу Кауэра с частотной характеристикой Чебышёва. Для оценки результатов синтеза и расчёта элементов разработана *JavaScript* программа. Показано, что для получения достоверных результатов число знаков в мантиссе переменных не может быть меньше порядка лестничной цепи. Для повышения точности расчётов рекомендуется применение математики произвольной или многократной точности. Для получения 15 достоверных знаков после запятой для фильтра 60−го порядка потребовалась длина мантиссы переменной g_n 120 знаков, а для расчёта промежуточных переменных в рядах Маклорена при расчёте тригонометрических, гиперболических и логарифмических функций – 140 знаков. Апробирование программы производилось на персональном компьютере с двухъядерным процессором *Intel*(*R*) *Core*(*TM*) *i*3−2100 *CPU* @3.10*GHz*, ОЗУ 3.41 ГБ и операционной системой *Windows XP Professional*. С увеличением порядка фильтра продолжительность расчёта возрастает практически в геометрической

прогрессии. Продолжительность расчёта фильтра 60–го порядка при 800 точках на графиках составила до 10 минут, а расчёт фильтра 20–го порядка занял 60 секунд. При 200 точках на графиках время расчёта для n=20 и n=60 составило 20 секунд и 3 минуты, соответственно.

Литература

1. Matthaei G.L. Tables of Chebyshev Impedance Transforming Networks of Low–Pass Filter Form // Proceedings of the IEEE. Vol. 52. Issue 8. Aug. 1964. P**.** 939 – 963.
2. Яковенко В.А. Аналитический расчет согласующих цепей лестничной структуры – URL: http://docplayer.ru/42908967-Analiticheskiy-raschet-soglasuyushchih-cepey-lestnichnoy-struktury.html
3. Zhu Y.S., Chen W.K. Low–pass impedance transformation networks // IEE Proc.–G Circuits Devices Syst. Vol. 144. No. 5. Oct. 1997. P. 284–288.
4. Zhu Y.S., Chen W.K. Computer Aided Design Of Communication Networks. – World Scientific Pub Co Inc. – 2000. – 628 p.
5. Змий Б.Ф., Ананьев А.В. Синтез пассивных LC–фильтров высших порядков с повышением уровня напряжения // Телекоммуникации. – 2016. – № 11. – С. 2–9.
6. Богачев В.М. Синтез цепей связи для широкополосных усилителей. Под ред. С.М. Смольского. – М.: МЭИ, 1980, 100 с.
7. Passive (Lumped Element) Filter Module – URL: http://www.nuhertz.com/software/software-modules/passive-lumped-element-filter
8. The GNU Multiple Precision Arithmetic Library – URL: http://www.wikiwand.com/en/GNU_Multiple_Precision_Arithmetic_Library
9. BCMath Library for JavaScript– URL: https://sourceforge.net/projects/bcmath–js/
10. A JavaScript library for arbitrary–precision decimal and non–decimal arithmetic – URL: https://github.com/MikeMcl/bignumber.js