VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Katedra elektrotechniky a informatiky

Program pro výpočet přenosu v semisymbolickém tvaru

Bakalářská práce

Autor: Jindřich Zerzánek

Vedoucí práce: Ing. Bohumil Brtník, Ph.D.

Jihlava 2011

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Jindřich Zerzánek**
Studijní program:	Elektrotechnika a informatika
Obor:	Počítačové systémy
Název práce:	**Program pro výpočet přenosu v semisymbolickém tvaru**
Cíl práce:	Úkolem je napsat program, který bude obsahovat grafické nebo textové rozhraní, pomocí kterého bude moci uživatel nakreslit (zadat) schéma obvodu.Po spuštění výpočtu program nejprve zkontroluje zapojení obvodu, pokud je zapojení korektní tak vypočte přenos obvodu

Ing. Bohumil Brtník, Ph.D.
vedoucí bakalářské práce

Ing. David Matoušek
vedoucí katedry
Katedra elektrotechniky a informatiky

Anotace

Tato práce je zaměřena na vytvoření programu, který počítá přenos napětí v lineárním obvodě po zadání prvků a jejich pozic v obvodu. Řešení výpočtu se provádí pomocí semisymbolického numerického algoritmu kde výstupem je charakteristická rovnice. První část je teoretická, zde je popsán podrobný postup analýzy a operací potřebných k zjištění výsledku. Druhá část se zabývá samotnou realizací programu, kde se jedná o popis důležitých funkcí a procedur.

Klíčová slova

semisymbolický numerický algoritmus, vlastní čísla matice, Danilevského metoda, napěťový přenos

Annotation

This work is focused on created program, which calculate voltage transmission in linear circuit after the enter elements and them position. The solution calculation is done by semisymbolic numeric algorithm, there result is characteristic equation. First part is theoretic here is procedure description of analysis and operation, which need for find result. Next part deals of implementation program, here are description important functions and procedures.

Keywords

semisymbolic numeric algorithm, eigenvalues, method of Danilevsky, voltage transmission

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Bohumilu Brtníkovi, Dr. za rady, připomínky a čas, který mi věnoval. Dále bych chtěl poděkovat RNDr. Janě Borůvkové, Ph.D. za pomoc s výpočty a objasnění nejasností vzniklých při řešení. Na závěr i svým rodičům za veškerou podporu, která se mi dostala během celého studia.

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Souhlasím s umístěním bakalářské práce v knihovně VŠPJ a s jejím užitím k výuce nebo k vlastní vnitřní potřebě VŠPJ.

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne

.....

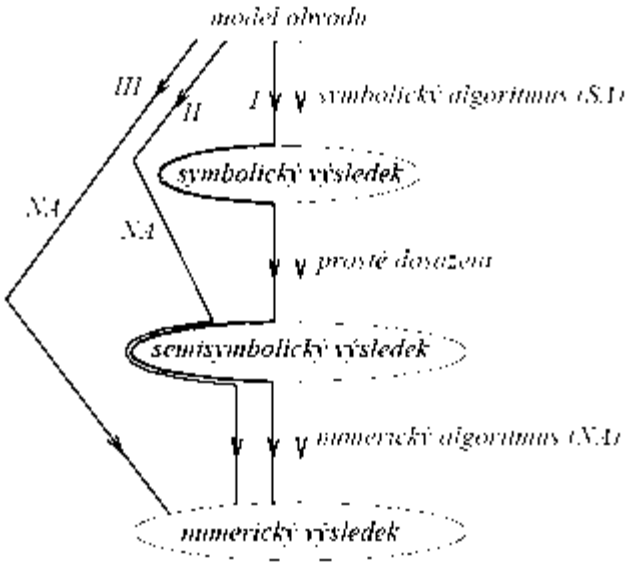
Podpis

Obsah

1 Úvod

Téma vytvoření programu pro výpočet napěťového přenosu v semisymbolickém tvaru jsem si vybral z tohoto důvodu, jedná se totiž o práci, která zahrnuje několik předmětů, se kterými jsem se setkal během studia. K dosažení výsledků mé bakalářské práce jsou třeba znalosti z lineární algebry, elektroniky, matematiky a na závěr i z programování. Na výběr byla i jiná témata zaměřená na tvorbu webových systémů, ale tyto práce se mi zdáli jednostranné. Jelikož jsem studoval obory s kombinací elektro-počítačů, chtěl jsem práci, která je zaměřena tímto směrem.

Mým cílem v této práci je naprogramovat aplikaci využívající semisymbolický numerický algoritmus, popsaný v příloze magazínu Slaboproudý obzor. Jeho nevýhoda podle bližších informací je v omezeném použití pro malé množství obvodů. Tuto metodu nelze použít na obvody, které neobsahují frekvenčně závislé prvky. Naopak výhodou této metody je přímý postup k výpočtu semisymbolického výsledku (tvar charakteristické rovnice, kde neznámá je Laplacův operátor), aniž by byl potřebný symbolický výsledek. Vztahy mezi formami výsledků analýzy a algoritmy analýzy jsou přehledně uvedeny na **Obr. 1**. [1]



Obr. 1: Typické tři cesty k výpočtu přenosu, moje metoda je II.

Program je složen ze samostatné aplikace, kde se zadávají prvky a dále knihoven s funkcemi potřebnými k výpočtu. Tyto knihovny obsahují maticové funkce jako

například součin matic, determinant matice, nebo inverzi. Mezi další funkce patří samotný výpočet zadaného algoritmu, rozdělený do fází.

Jako programovací jazyk používám C# jehož syntaxe vychází z jazyka C++, který obsahuje velké množství výhod, a zdokonaleních na rozdíl od svého předchůdce. Hlavní důvod mého výběru je, že jsem si chtěl rozšířit své znalosti, jedná se o objektově orientovaný jazyk a programy vytvořené v něm, je možné lehce přenášet mezi operačními systémy Windows na kterých je nainstalován .NET Framework 3.5.

Význam této práce spočívá v otestování této metody. Program bude užit pro ověření omezujících vlastností tohoto algoritmu ve výuce analogových systémů nebo dalších předmětech s tímto zaměřením.

2 Rozbor zadání a cílů

2.1 Rozbor

Vývojem programů na navrhování a počítání obvodů pomocí počítače se zabývalo již mnoho odborníků jak v minulosti tak i nyní. Existuje spousty analýz a metod, které vedou k výsledku, jenž nás zajímá. Úkolem této práce je naprogramovat aplikaci, která řeší výpočet přenosu napětí, vstupní a výstupní impedanci v lineárním obvodě. K dosažení výsledků vedou tři typické cesty, jak jsem již uvedl v úvodu. Podrobně se jimi zabývá profesor Ing. Dalibor Biolek, CSc. z Vojenské akademie v Brně ve skriptech „Navrhování elektronických obvodů počítačem“. Zjištění námi požadovaného výsledku lze prostřednictvím symbolických (tvar, ve kterém jsou symboly parametrů prvků spolu s Laplacovým operátorem)[2] nebo semisymbolických výsledků existují i metody, které počítají složité soustavy rovnic a získají tak přímo numerický výsledek. Jelikož cílem práce není zjištění numerického výsledku, ale jen semisymbolického, postačí k řešení jedna z možných analýz, které jsou vidět na **Obr. 1**. První možností je vyhodnotit symbolický výsledek, kde jeho symboly prvků nahradíme hodnotami. Druhou možností je přímý výpočet přes zjišťování vlastních čísel vodivostní a imaginární matice, k tomu slouží semisymbolický numerický algoritmus.

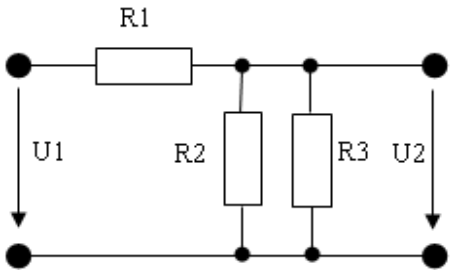
2.2 Cíle

- Teoretický postup výpočtu napěťového přenosu, vstupní a výstupní impedance.
- Návrh funkcí a algoritmů potřebných k výpočtu.
- Model a rozložení aplikace
- Vytvoření testovací konsolové verze.
- Vytvoření finální grafické aplikace.
- Otestování a porovnání výsledků s programem SNAP.

2.3 Pojmy

2.3.1 Symbolická analýza

Tento způsob řešení využívá složité postupy, které vedou k výsledku se symboly hodnot prvků a Laplacovým operátorem (pokud obsahují frekvenčně závislé prvky). Analýza slouží jako základ pro ostatní formy výsledků, stačí jen dosadit skutečné hodnoty a tím se získají semisymbolické i numerické výsledky. Algoritmus této metody je velmi náročný, protože manipuluje s proměnnými a s každým dalším prvkem složitost exponenciálně roste. Příkladem řešící symbolickou analýzu je COCO algoritmus, jehož metoda spočívá v rozvoji algebraického doplňku.[2]

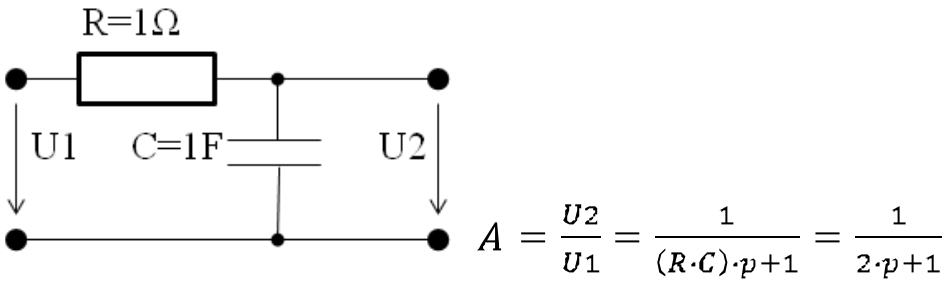


$$A = \frac{U2}{U1} = \frac{R2 \cdot R3}{R1 \cdot R3 + R2 \cdot R3 + R1 \cdot R2}$$

Obr. 2: Ukázka schématu obvodu s výpočtem přenosu napětí v symbolickém tvaru.

2.3.2 Semisymbolická analýza

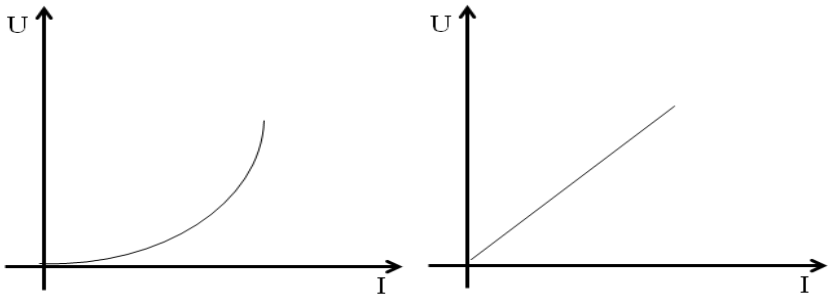
Jedná se o analýzu s výsledkem ve formě charakteristické rovnice, kde neznámá je Laplacův operátor, který se pro výpočet do numerického tvaru nahrazuje $j \cdot \omega$. Zjistit semisymbolický výsledek lze dvěma způsoby, jak jsem popsal výše, bohužel každý z nich má své výhody a nevýhody. První způsob vede přes symbolickou metodu, kde se nahradí symboly číselnými hodnotami prvků, tato cesta je jednoduchá, ale obsahuje nevýhody, které vznikly už u symbolické metody. Druhým způsobem je přímý výpočet přes semisymbolický numerický algoritmus, který vypočítává koeficienty rovnice přes vlastní čísla. Tento způsob má výhodu v tom, že je rychlejší pro rozsáhlejší obvody oproti prvnímu, řešení probíhá úpravou matic a po té se pomocí Danilevského metody zjistí koeficienty rovnice. Nevýhoda tohoto způsobu spočívá v tom, že je lze použít jen u malého množství obvodů.[2]



Obr. 3: Ukázka schématu RC článku s výpočtem přenosu napětí v semisymbolickém tvaru.

2.3.3 Lineární obvod

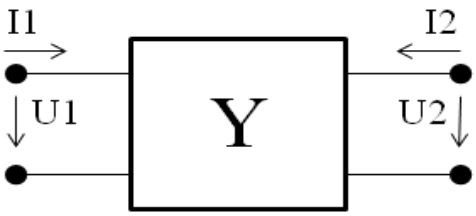
Je obvod složený z obvodových prvků, jejichž parametry jsou nezávislé, tedy nejsou ovlivňovány na procházejícím proudu nebo na ně přiloženém napětí. Typický lineární obvod je složený z rezistorů či kondenzátorů, v nelineárních obvodech nalezneme naopak prvky typu dioda nebo tranzistor. Rozdíly v linearitách jsou nejlépe vidět na těchto charakteristikách **Obr. 4**.[5]



Obr. 4: Levá charakteristika znázorňuje nelineární prvek a pravá lineární.

2.3.4 Napěťový přenos, vstupní a výstupní impedance

Je znám i pod názvem napěťové zesílení, jedná se o poměr výstupního napětí ku vstupnímu, abychom se dostali k těmto napětím vysvětlíme si podrobněji postup na dvojbranu.[3]



Obr. 5: Dvojbran

Na **Obr. 5** je vidět dvojbran který lze popsat těmito rovnicemi:

$$I_1 = y_{11} \cdot U_1 + y_{12} \cdot U_2$$
$$I_2 = y_{21} \cdot U_1 + y_{22} \cdot U_2$$

Kde y_{11} je vstupní admitance při výstupu na krátko, y_{12} zpětně přenosová admitance při vstupu na krátko, y_{21} přenosová admitance při výstupu na krátko a y_{22} výstupní admitance při vstupu na krátko. Jestliže je dvojbran nezatížený, tak $I_2 = 0$ z toho vyplývá, že rovnice budou vypadat takto:

$$0 = y_{21} \cdot U_1 + y_{22} \cdot U_2$$

Lze tedy vyjádřit $y_{11} \cdot U_1$ a získat toto:

$$-y_{21} \cdot U_1 = y_{22} \cdot U_2$$

Finální úprava vede k získání základního výpočtu přenosu:

$$A_u = \frac{U_2}{U_1} = -\frac{y_{21}}{y_{22}}$$

Každý dvojbran má vnitřní zapojení, což je vlastně námi zadaný obvod, u kterého zjišťujeme zesílení. Pro výpočet je třeba redukce vnitřních uzlů, to jsou takové, co nejsou spojené s vnějšími vstupy. Následně se vytvoří matice o rozměrech zredukovaných počtů uzlů, tedy pokud je obvod se třemi uzly a vnějších vývodů se dotýkají pouze uzly 1 a 2, bude matice o rozměrech 2×2. Výsledný přenos je tedy $-\frac{y_{21}}{y_{22}}$.

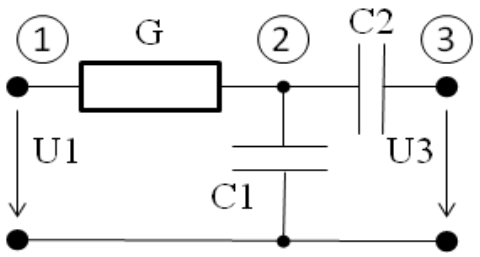
Vstupní a výstupní impedance se řeší při odstranění všech nezávislých zdrojů proudu a napětí u daného vstupního nebo výstupního uzlu. Lze je tedy charakterizovat těmito rovnicemi[3]:

$$Z_{in} = \frac{U_1}{I_1} \qquad Z_{out} = \frac{U_2}{I_2}$$

3 Popis výpočtu a návrh programu

3.1 Metoda uzlových napětí

Princip této metody je velmi jednoduchý, vysvětlení provedu na daném obvodu **Obr. 6**. Schéma obsahuje tři pasivní prvky, rezistor s vodivostí G a dva kapacitory C1, C2.[6]



Obr. 6: Obvod s vyznačenými uzly.

Ve schématu jsou vyznačeny uzly mezi jednotlivými prvky. Dalším krokem je sestavení rovnic k jednotlivým uzlům dle I. Kirchhoffova zákona $\sum I = \sum Y \cdot U$.

První uzel: $0 = G \cdot (U1 - U2)$

Druhý uzel: $0 = G \cdot (U2 - U1) + C1 \cdot j\omega \cdot U2 + C2 \cdot j\omega \cdot (U2 - U3)$

Třetí uzel: $0 = C2 \cdot j\omega \cdot (U3 - U2)$

Tento tvar je bohužel nevyhovující, musí se tedy rovnice upravit roznásobením.

$$0 = G \cdot U1 - G \cdot U2$$
$$0 = G \cdot U2 - G \cdot U1 + C1 \cdot j\omega \cdot U2 + C2 \cdot j\omega \cdot U2 - C2 \cdot j\omega \cdot U3$$
$$0 = C2 \cdot j\omega \cdot U3 - C2 \cdot j\omega \cdot U2$$

Nyní je třeba vytknout napětí všude, kde to lze.

Nelze: $0 = G \cdot U1 - G \cdot U2$

Lze: $0 = -G \cdot U1 + U2 \cdot (G + C1 \cdot j\omega + C2 \cdot j\omega) - C2 \cdot j\omega \cdot U3$

Nelze: $0 = C2 \cdot j\omega \cdot U3 - C2 \cdot j\omega \cdot U2$

V tuto chvíli je možné přepsat rovnice do maticového.

$$\begin{pmatrix} G & -G & 0 \\ -G & G + C1 \cdot j\omega + C2 \cdot j\omega & -C2 \cdot j\omega \\ 0 & -C2 \cdot j\omega & C2 \cdot j\omega \end{pmatrix} \cdot \begin{pmatrix} U1 \\ U2 \\ U3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Která odpovídá symbolickému tvaru $Y \cdot U = I$

3.2 Teoretický výpočet vlastních čísel matice

V předešlé kapitole je vysvětleno jak se dostat k admitanční matici potřebnou pro zjištění vlastních čísel, rozepíšeme ji na součet dvou reálných matic. Vodivostní G, jenž obsahuje reálné složky a susceptanční B obsahující imaginární složky, která je násobena Laplacovým operátorem *p*.[3]

$$Y = G + p \cdot B$$

Pro výpočet přenosu napětí, vstupní nebo výstupní impedance je třeba vyjádřit binominální determinanty, jakožto mnohočleny proměnné $p=j\omega$.

$$A = \frac{U_2}{U_1} = \frac{\Delta_{1;2}}{\Delta_{1;1}} \qquad Z_{in} = \frac{U_1}{I_1} = \frac{\Delta_{1;1}}{\Delta} \qquad Z_{out} = \frac{U_2}{I_2} = \frac{\Delta_{2;2}}{\Delta}$$

Vyjádříme tedy Δ=det(G+p·B), tento tvar lze upravit tímto postupem:

$$\begin{aligned} \Delta = \det(G + p \cdot B) &= \det[B \cdot (B^{-1} \cdot G + p \cdot B^{-1} \cdot B)] \\ &= \det(B) \cdot \det(B^{-1} \cdot G + p \cdot 1) \end{aligned}$$

Následuje zjednodušení úlohy, tak že bude $C = \det(B)$ násobná konstanta a $\det(X + p \cdot 1)$ koeficienty polynomu. Této proceduře se nazývá deflace obvodových rovnic.

$$C \cdot \det(X + p \cdot 1)$$

Nyní vznikne klasická úloha vlastních čísel:

$$\det(X + p \cdot 1) = 0$$

Vyhodnocování charakteristických čísel probíhá klasicky dle tvaru (A-λ·E)·x=0 což představuje soustavu rovnic:

$$\begin{pmatrix} (a_{11}-\lambda)\cdot x_1 & + & a_{12}\cdot x_2 & + & \cdots & + & a_{1n}\cdot x_n \\ a_{21}\cdot x_1 & + & (a_{22}-\lambda)\cdot x_2 & + & \cdots & + & a_{2n}\cdot x_n \\ \vdots & & \vdots & & \cdots & & \vdots \\ a_{n1}\cdot x_1 & + & a_{n2}\cdot x_2 & + & \cdots & + & (a_{nn}-\lambda)\cdot x_n \end{pmatrix} = \begin{matrix} 0 \\ 0 \\ \vdots \\ 0 \end{matrix}$$

Tyto rovnice se řeší použitím posloupnosti podobnostních transformací neboli Danilevského metody na tvar:

$$X_{n+1} = T^{-1}\cdot X_n \cdot T$$

X je původní matice, ke které chceme zjistit vlastní čísla. Inverzní *T* se zvolí, poté se k němu provede inverze, tím vznikne matice *T*. Po součinu těchto tří čtvercových matic si výsledná zachová vlastní čísla z původní. Tento tvar se nazývá Frobeninův, jeho tvar má v prvním řádku matice koeficienty charakteristické rovnice, zbylá část od druhého řádku vypadá takto, po diagonále jsou 1 a zbytek 0.

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Poněvadž původní tvar byl *det(X+p·1)* musí se tedy kořeny zjistit pomocí determinantů a bude to vypadat následovně:

$|x_1|=x_1$ $\qquad \begin{vmatrix} x_1 & x_2 \\ 1 & 0 \end{vmatrix} = x_1\cdot 0 - x_2\cdot 1 = -x_2 \qquad \begin{vmatrix} x_1 & x_2 & x_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix} = x_3\cdot 1\cdot 1 = x_3$

$$\det(X + p\cdot 1) = p^3 + x_1\cdot p^2 - x_2\cdot p + x_3$$

Jak jsem již zmiňoval v předešlých kapitolách, u této metody dochází k možnosti, že nelze aplikovat tuto analýzu na většinu obvodů, poněvadž matice *B* nebo i G vyjde jako singulární a nelze k ní udělat inverzi, tento stav může nastat i při úpravách matice *X*.[3]

Ukázka stanovení koeficientů pomocí Danilevského metody:

1. Matice X:

$$X_0 = \begin{pmatrix} 2 & -2 \\ -2 & 4 \end{pmatrix}$$

2. Určíme T^{-1} tak, že vezmeme řádek z matice X a zbytek bude odpovídat jedničkové matici:

$$T^{-1} = \begin{pmatrix} -2 & 4 \\ 0 & 1 \end{pmatrix}$$

3. Vypočítáme T pomocí inverze k matici T^{-1}:

$$det\begin{pmatrix}-2 & 4\\ 0 & 1\end{pmatrix} = -2 \Rightarrow T = \begin{pmatrix}\frac{(-1)^{1+1}\cdot 1}{-2} & \frac{(-1)^{1+2}\cdot 4}{-2}\\ \frac{(-1)^{2+1}\cdot 0}{-2} & \frac{(-1)^{2+2}\cdot -2}{-2}\end{pmatrix} = \begin{pmatrix}-\frac{1}{2} & 2\\ 0 & 1\end{pmatrix}$$

4. Nyní se provede součin těchto tří matic pro dosažení Frobeninova tvaru:

$$X_{(1)} = T^{-1}\cdot X_{(0)}\cdot T$$

$$X_{(1)} = \begin{pmatrix}-2 & 4\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}2 & -2\\ -2 & 4\end{pmatrix}\cdot\begin{pmatrix}-\frac{1}{2} & 2\\ 0 & 1\end{pmatrix}$$

$$X_{(1)} = \begin{pmatrix}6 & -4\\ 1 & 0\end{pmatrix}$$

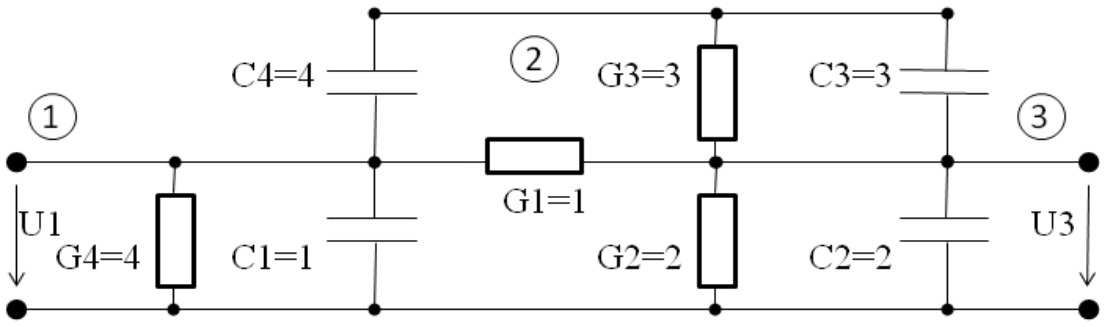
5. Zjištění koeficientů pomocí determinantů:

$|x_1|=|6|=6$ $\qquad x_2 = \begin{vmatrix}6 & -4\\ 1 & 0\end{vmatrix} = 6\cdot 0 - (-4)\cdot 1 = 4$

$$\lambda^2 + x_1\cdot\lambda + x_2 = \lambda^2 + 6\lambda + 4$$

3.3 Názorný příklad analýzy semisymbolickou metodou

Zadaný je následující obvod **Obr. 7**, ke kterému se provede výpočet přenosu, impedance vstupní i výstupní. Je zde možnost, že nemusí vyjít všechny hledané výsledky.



Obr. 7: Zadané schéma pro názorný výpočet, hodnoty prvků odpovídají základním veličinám.

Obvod obsahuje tři uzly,tedy matice budou mít tedy rozměr 3×3. Nyní se vytvoří admitanční matice s dosazením hodnot prvků.

$$Y = G + p \cdot B =$$

$$= \begin{matrix} 1: \\ 2: \\ 3: \end{matrix} \begin{pmatrix} G1+G4 & 0 & -G1 \\ 0 & G3 & -G3 \\ -G1 & -G3 & G1+G3+G2 \end{pmatrix} + p \cdot \begin{matrix} 1: \\ 2: \\ 3: \end{matrix} \begin{pmatrix} C4+C1 & -C4 & 0 \\ -C4 & C4+C3 & -C3 \\ 0 & -C3 & C3+C2 \end{pmatrix} =$$

$$= \begin{pmatrix} 5 & 0 & -1 \\ 0 & 3 & -3 \\ -1 & -3 & 6 \end{pmatrix} + p \cdot \begin{pmatrix} 5 & -4 & 0 \\ -4 & 7 & -3 \\ 0 & -3 & 5 \end{pmatrix}$$

Admitanční matice je nyní připravena pro výpočty, přenosu a impedancí.

$$A = \frac{U_3}{U_1} = \frac{\Delta_{1;3}}{\Delta_{1;1}}$$

$$Z_{in} = \frac{\Delta_{1;1}}{\Delta}$$

$$Z_{out} = \frac{\Delta_{3;3}}{\Delta}$$

Je třeba vyřešit postupně potřebné determinanty, viz teoretický výpočet $C \cdot det(B^{-1} \cdot G + p \cdot 1)$.

- Výpočet $\Delta_{1;1}$

$$\Delta_{1;1} = \begin{pmatrix} 3 & -3 \\ -3 & 6 \end{pmatrix} + p \cdot \begin{pmatrix} 7 & -3 \\ -3 & 5 \end{pmatrix} = G + p \cdot B$$

$$C = det \begin{pmatrix} 7 & -3 \\ -3 & 5 \end{pmatrix} = 7 \cdot 5 - (-3 \cdot -3) = 35 - 9 = 26$$

$$B^{-1} = inverse \begin{pmatrix} 7 & -3 \\ -3 & 5 \end{pmatrix} = \begin{pmatrix} \frac{5}{26} & \frac{3}{26} \\ \frac{3}{26} & \frac{7}{26} \end{pmatrix}$$

$$C \cdot \det(B^{-1} \cdot G + p \cdot 1) = C \cdot \det(X + p \cdot 1)$$

$$X = \begin{pmatrix} \frac{5}{26} & \frac{3}{26} \\ \frac{3}{26} & \frac{7}{26} \end{pmatrix} \cdot \begin{pmatrix} 3 & -3 \\ -3 & 6 \end{pmatrix} = \begin{pmatrix} \frac{3}{13} & \frac{3}{26} \\ -\frac{6}{13} & \frac{33}{26} \end{pmatrix}$$

$$X_n = T^{-1} \cdot X \cdot T$$

$$T = inverse\begin{pmatrix} -\frac{6}{13} & \frac{33}{26} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -\frac{13}{6} & \frac{11}{4} \\ 0 & 1 \end{pmatrix}$$

$$X_1 = \begin{pmatrix} -\frac{6}{13} & \frac{33}{26} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{3}{13} & \frac{3}{26} \\ -\frac{6}{13} & \frac{33}{26} \end{pmatrix} \cdot \begin{pmatrix} -\frac{13}{6} & \frac{11}{4} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} & -\frac{9}{26} \\ 1 & 0 \end{pmatrix}$$

$$\left|\frac{3}{2}\right| = \frac{3}{2} = 1{,}5 \quad \left|\begin{matrix} \frac{3}{2} & -\frac{9}{26} \\ 1 & 0 \end{matrix}\right| = \frac{3}{2} \cdot 0 - \frac{9}{26} \cdot 1 = \frac{9}{26} = 0{,}346$$

$$C \cdot \det(X + p \cdot 1) = 26 \cdot (p^2 + 1{,}5 \cdot p + 0{,}346)$$

- Výpočet $\Delta_{1;3}$

$$\Delta_{1;3} = \begin{pmatrix} 0 & 3 \\ -1 & -3 \end{pmatrix} + p \cdot \begin{pmatrix} -4 & 7 \\ 0 & -3 \end{pmatrix} = G + p \cdot B$$

$$C = det\begin{pmatrix} -4 & 7 \\ 0 & -3 \end{pmatrix} = -4 \cdot -3 - 7 \cdot 0 = 12$$

$$B^{-1} = inverse\begin{pmatrix} -4 & 7 \\ 0 & -3 \end{pmatrix} = \begin{pmatrix} -\frac{1}{4} & -\frac{7}{12} \\ 0 & -\frac{1}{3} \end{pmatrix}$$

$$C \cdot \det(B^{-1} \cdot G + p \cdot 1) = C \cdot \det(X + p \cdot 1)$$

$$X = \begin{pmatrix} -\frac{1}{4} & -\frac{7}{12} \\ 0 & -\frac{1}{3} \end{pmatrix} \cdot \begin{pmatrix} 0 & 3 \\ -1 & -3 \end{pmatrix} = \begin{pmatrix} \frac{7}{12} & 1 \\ \frac{1}{3} & 1 \end{pmatrix}$$

$$X_n = T^{-1} \cdot X \cdot T$$

$$T = inverse\begin{pmatrix} \frac{1}{3} & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & -3 \\ 0 & 1 \end{pmatrix}$$

$$X_1 = \begin{pmatrix} \frac{1}{3} & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{7}{12} & 1 \\ \frac{1}{3} & 1 \end{pmatrix} \cdot \begin{pmatrix} 3 & -3 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{19}{12} & -\frac{1}{4} \\ 1 & 0 \end{pmatrix}$$

$$\left|\frac{19}{12}\right| = \frac{19}{12} = 1{,}583 \quad \left|\begin{matrix} \frac{19}{12} & -\frac{1}{4} \\ 1 & 0 \end{matrix}\right| = \frac{19}{12} \cdot 0 - \left(-\frac{1}{4} \cdot 1\right) = \frac{1}{4} = 0{,}25$$

$$C \cdot \det(X + p \cdot 1) = 12 \cdot (p^2 + 1{,}583 \cdot p + 0{,}25)$$

- Výpočet $\Delta_{3;3}$

$$\Delta_{3;3} = \begin{pmatrix} 5 & 0 \\ 0 & 3 \end{pmatrix} + p \cdot \begin{pmatrix} 5 & -4 \\ -4 & 7 \end{pmatrix} = G + p \cdot B$$

$$C = det \begin{pmatrix} 5 & -4 \\ -4 & 7 \end{pmatrix} = 5 \cdot 7 - (-4 \cdot -4) = 35 - 16 = 9$$

$$B^{-1} = inverse \begin{pmatrix} 5 & -4 \\ -4 & 7 \end{pmatrix} = \begin{pmatrix} \frac{7}{19} & \frac{4}{19} \\ \frac{4}{19} & \frac{5}{19} \end{pmatrix}$$

$$C \cdot \det(B^{-1} \cdot G + p \cdot 1) = C \cdot \det(X + p \cdot 1)$$

$$X = \begin{pmatrix} \frac{7}{19} & \frac{4}{19} \\ \frac{4}{19} & \frac{5}{19} \end{pmatrix} \cdot \begin{pmatrix} 5 & 0 \\ 0 & 3 \end{pmatrix} = \begin{pmatrix} \frac{35}{19} & \frac{12}{19} \\ \frac{20}{19} & \frac{15}{19} \end{pmatrix}$$

$$X_n = T^{-1} \cdot X \cdot T$$

$$T = inverse \begin{pmatrix} \frac{20}{19} & \frac{15}{19} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{19}{20} & -\frac{3}{4} \\ 0 & 1 \end{pmatrix}$$

$$X_1 = \begin{pmatrix} \frac{20}{19} & \frac{15}{19} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{35}{19} & \frac{12}{19} \\ \frac{20}{19} & \frac{15}{19} \end{pmatrix} \cdot \begin{pmatrix} \frac{19}{20} & -\frac{3}{4} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{50}{19} & -\frac{15}{19} \\ 1 & 0 \end{pmatrix}$$

$$\left|\frac{50}{19}\right| = \frac{50}{19} = 2{,}632 \qquad \begin{vmatrix} \frac{50}{19} & -\frac{15}{19} \\ 1 & 0 \end{vmatrix} = \frac{50}{19} \cdot 0 - \left(-\frac{15}{19} \cdot 1\right) = \frac{15}{19} = 0{,}789$$

$$C \cdot \det(X + p \cdot 1) = 19 \cdot (p^2 + 2{,}632 \cdot p + 0{,}789)$$

- Výpočet Δ

$$\Delta = \begin{pmatrix} 5 & 0 & -1 \\ 0 & 3 & -3 \\ -1 & -3 & 6 \end{pmatrix} + p \cdot \begin{pmatrix} 5 & -4 & 0 \\ -4 & 7 & -3 \\ 0 & -3 & 5 \end{pmatrix} = G + p \cdot B$$

$$C = det \begin{pmatrix} 5 & -4 & 0 \\ -4 & 7 & -3 \\ 0 & -3 & 5 \end{pmatrix} = 50$$

$$B^{-1} = inverse\begin{pmatrix} 5 & -4 & 0 \\ -4 & 7 & -3 \\ 0 & -3 & 5 \end{pmatrix} = \begin{pmatrix} \frac{13}{25} & \frac{2}{5} & \frac{6}{25} \\ \frac{2}{5} & \frac{1}{2} & \frac{3}{10} \\ \frac{6}{25} & \frac{3}{10} & \frac{19}{50} \end{pmatrix}$$

$$C \cdot \det(B^{-1} \cdot G + p \cdot 1) = C \cdot \det(X + p \cdot 1)$$

$$X = \begin{pmatrix} \frac{13}{25} & \frac{2}{5} & \frac{6}{25} \\ \frac{2}{5} & \frac{1}{2} & \frac{3}{10} \\ \frac{6}{25} & \frac{3}{10} & \frac{19}{50} \end{pmatrix} \cdot \begin{pmatrix} 5 & 0 & -1 \\ 0 & 3 & -3 \\ -1 & -3 & 6 \end{pmatrix} = \begin{pmatrix} \frac{59}{25} & \frac{12}{25} & -\frac{7}{25} \\ \frac{17}{10} & \frac{3}{5} & -\frac{1}{10} \\ \frac{41}{50} & -\frac{6}{25} & \frac{57}{50} \end{pmatrix}$$

$$X_n = T^{-1} \cdot X \cdot T$$

$$T = inverse\begin{pmatrix} 1 & 0 & 0 \\ \frac{41}{50} & -\frac{6}{25} & \frac{57}{50} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{41}{12} & -\frac{25}{6} & \frac{19}{4} \\ 0 & 0 & 1 \end{pmatrix}$$

$$X_1 = \begin{pmatrix} 1 & 0 & 0 \\ \frac{41}{50} & -\frac{6}{25} & \frac{57}{50} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{59}{25} & \frac{12}{25} & -\frac{7}{25} \\ \frac{17}{10} & \frac{3}{5} & -\frac{1}{10} \\ \frac{41}{50} & -\frac{6}{25} & \frac{57}{50} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ \frac{41}{12} & -\frac{25}{6} & \frac{19}{4} \\ 0 & 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 4 & -2 & 2 \\ \frac{119}{50} & \frac{1}{10} & \frac{49}{50} \\ 0 & 1 & 0 \end{pmatrix}$$

Tento tvar neodpovídá Frobeninovu, proto se musí dále upravit Danilevského metodou.

$$X_n = T^{-1} \cdot X \cdot T$$

$$T = inverse\begin{pmatrix} \frac{119}{50} & \frac{1}{10} & \frac{49}{50} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{50}{119} & -\frac{5}{119} & -\frac{7}{17} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$X_2 = \begin{pmatrix} \frac{119}{50} & \frac{1}{10} & \frac{49}{50} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{4}{119} & -\frac{2}{1} & \frac{2}{49} \\ \frac{50}{50} & \frac{10}{10} & \frac{50}{50} \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{50}{119} & -\frac{5}{119} & -\frac{7}{17} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{41}{10} & -\frac{209}{50} & \frac{21}{25} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\left|\frac{41}{10}\right| = \frac{41}{10} = 4{,}1 \qquad \begin{vmatrix} \frac{41}{10} & -\frac{209}{50} \\ 1 & 0 \end{vmatrix} = \frac{41}{10} \cdot 0 - \left(-\frac{209}{50} \cdot 1\right) = \frac{209}{50} = 4{,}18$$

$$\begin{vmatrix} \frac{41}{10} & -\frac{209}{50} & \frac{21}{25} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix} = \frac{21}{25} = 0{,}84$$

$$C \cdot \det(X + p \cdot 1) = 50 \cdot (p^3 + 4{,}1 \cdot p^2 + 4{,}18 \cdot p + 0{,}84)$$

Nyní se vyhodnotí přímo výsledky semisymbolické analýzy, které jsme chtěli zjistit, charakteristické rovnice jsme si připravili a teď se pouze dosadí.

$$A = \frac{U_3}{U_1} = \frac{\Delta_{1;3}}{\Delta_{1;1}} = \frac{12 \cdot (p^2 + 1{,}583 \cdot p + 0{,}25)}{26 \cdot (p^2 + 1{,}5 \cdot p + 0{,}346)}$$

$$Z_{in} = \frac{\Delta_{1;1}}{\Delta} = \frac{26 \cdot (p^2 + 1{,}5 \cdot p + 0{,}346)}{50 \cdot (p^3 + 4{,}1 \cdot p^2 + 4{,}18 \cdot p + 0{,}84)}$$

$$Z_{out} = \frac{\Delta_{3;3}}{\Delta} = \frac{19 \cdot (p^2 + 2{,}632 \cdot p + 0{,}789)}{50 \cdot (p^3 + 4{,}1 \cdot p^2 + 4{,}18 \cdot p + 0{,}84)}$$

3.4 Proč C#

Pro naprogramování aplikace jsem si vybral vysokoúrovňový objektově orientovaný programovací jazyk C# (C Sharp) od společnosti Microsoft. Tento jazyk má výhodu, že pracuje na platformě .NET Framework, která již dnes je součástí operačních systémů Windows Vista nebo 7, lze jej také doinstalovat i na starší systémy, jako například Windows XP jenž se i v dnešní době velmi používají. Tato technologie umožňuje přenositelnost naprogramované aplikace bez nutnosti nestandardních prerekvizit a její velikost i při použití formulářových prvků zabírá několik KB (kilobajtů).

3.4.1 Výhody C#

Zde jsou vypsány důležité výhody[4]:

- Jednoduchý, moderní, mnohoúčelový a objektově orientovaný programovací jazyk.
- Poskytuje podporu pro principy softwarového inženýrství, jako jsou: hlídání hranic polí, detekce použití neinicializovaných proměnných a automatický garbage collector.
- Jazyk je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích.
- Přenositelnost zdrojového kódu je velmi důležitá, obzvláště pro ty programátory, kteří jsou obeznámeni s C a C++.
- Je navržen pro psaní aplikací, jak pro zařízení se sofistikovanými operačními systémy, tak pro zařízení s omezenými možnostmi.
- Přestože by programy psané v C# neměly plýtvat s přiděleným procesorovým časem a pamětí, nemohou se měřit s aplikacemi psanými v C nebo jazyce symbolických adres.

3.4.2 Vlastnosti C#

Zde jsou vypsány důležité vlastnosti[4]:

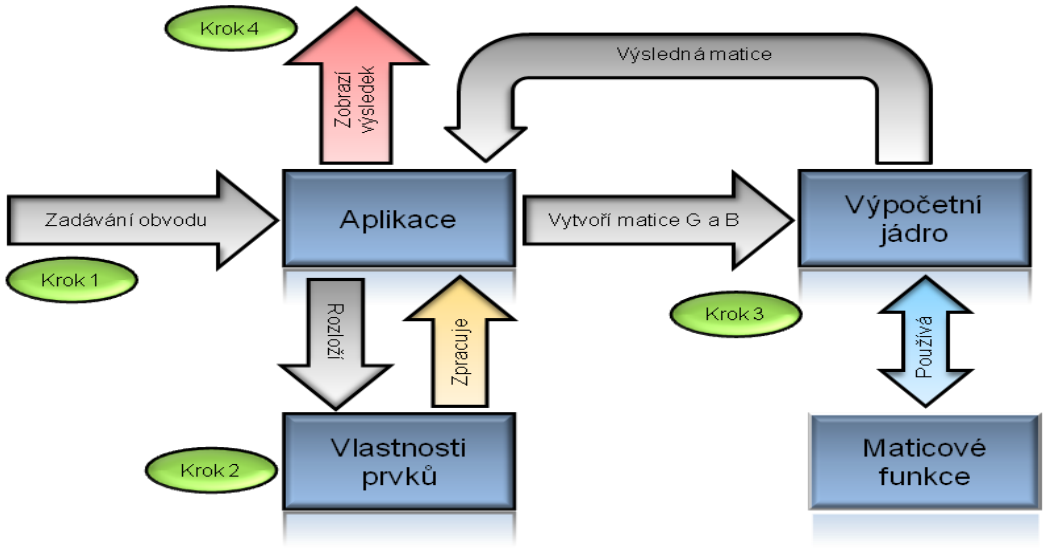
- Neexistuje vícenásobná dědičnost - to znamená, že každá třída může být potomkem pouze jedné třídy. Toto rozhodnutí bylo přijato, aby se předešlo komplikacím a přílišné složitosti, která je spojena s vícenásobnou dědičností. Třída může implementovat libovolný počet rozhraní.
- Neexistují žádné globální proměnné a metody. Všechny funkce a metody musí být deklarovány uvnitř tříd. Náhradou za ně jsou statické metody a proměnné veřejných tříd.
- V objektově orientovaném programování se z důvodu dodržení principu zapouzdření často používá vzor, kdy k datovým atributům třídy lze zvenčí

přistupovat pouze nepřímo a to pomocí dvou metod get (accessor) a set (mutator). V C# lze místo toho definovat tzv. Property, která zvenčí stále funguje jako datový atribut, ale uvnitř Property si můžeme definovat get a set metody. Výhodou je jednodušší práce s datovým atributem při zachování principu zapouzdření.

- Neobsahuje a ani nepotřebuje dopřednou deklaraci - není důležité pořadí deklarace metod.
- Je case sensitive - to znamená, že rozlišuje mezi velkými a malými písmeny. Identifikátory "hodnota" a "Hodnota" tedy nejsou na rozdíl od VB.NET ekvivalentní.

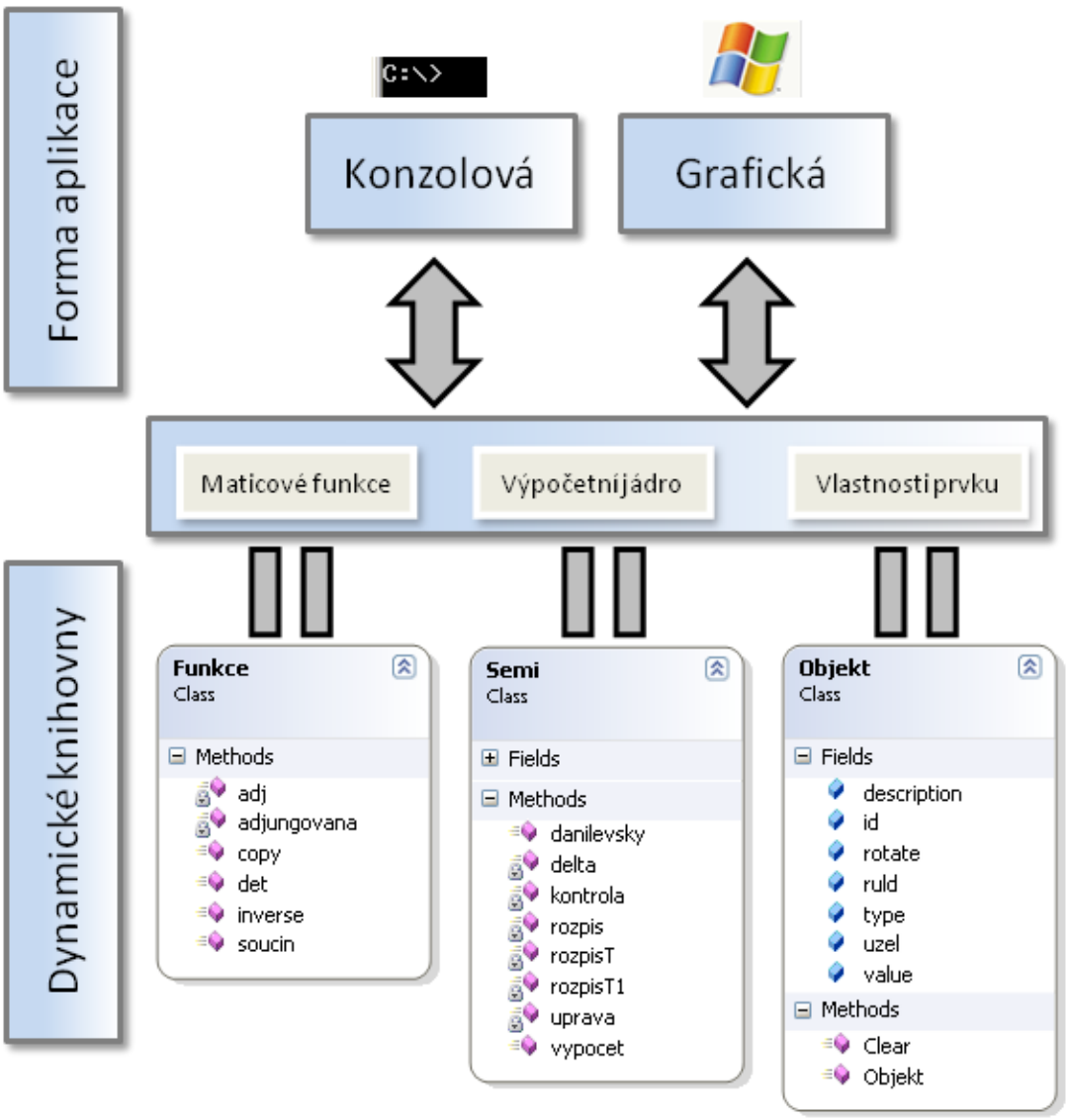
3.5 Implementace a návrh

V zadání stojí, že aplikace bude konzolového nebo grafického rozhraní. Proto jsem program navrhoval tak, aby bylo možné vyměnit možnosti rozhraní, oddělil jsem tedy funkce potřebné k výpočtu, samotné výpočetní jádro a vlastnosti prvků od samotné aplikace. Tyto oddělené části tvoří samostatné dynamické knihovny, lze jej pak jednoduše přidat k dané formě programu. Princip aplikace vysvětluje obrázek **Obr. 8**, jsou zde vyznačeny jednotlivé kroky.



Obr. 8: Zjednodušený princip programu.

Dle principu byla vytvořena následující implementace, která je používána ve finální verzi programu viz **Obr. 9**.



Obr. 9: Návrh implementace aplikace, s obsahem tříd.

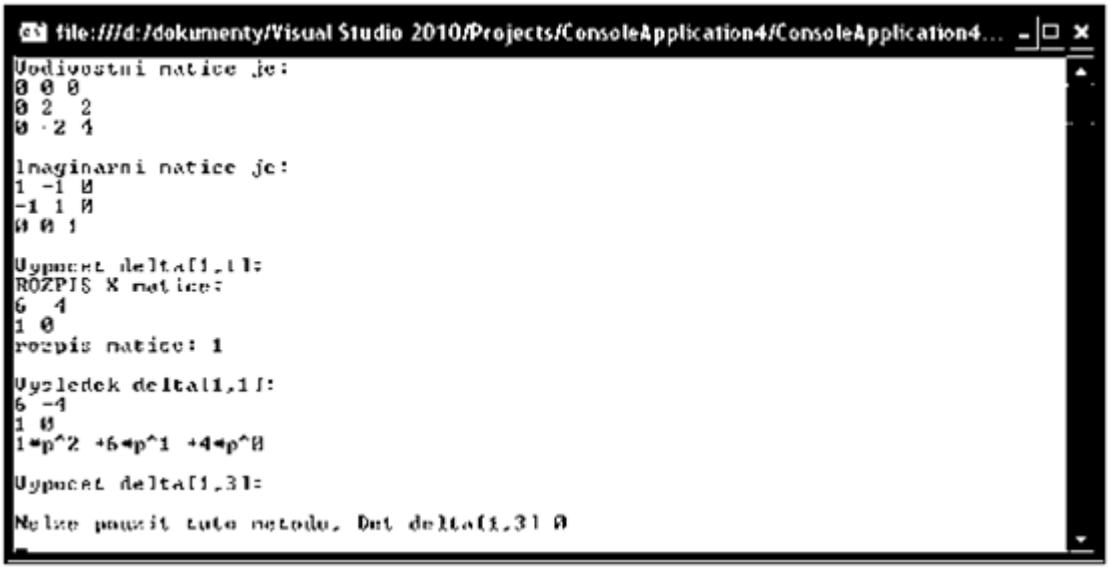
Jako první jsem vytvořil konzolovou verzi, která fungovala formou dialogu a sloužila pouze jako testovací. Uživatel zadal číslo odpovídající prvku, poté jeho umístění a hodnotu. Po dokončení zadávání se zobrazil výsledek.

Ukázka zadání testovacího příkladu Wienův článek. Matice je 3×3 a uzel 4 je zem, prvky jsou zadané takto:

1. C1 pozice 1,2 velikost 1F.
2. G1 pozice 2,3 velikost 2S.

3. C2 pozice 3,4 velikost 1F.
4. G2 pozice 3,4 velikost 2S.

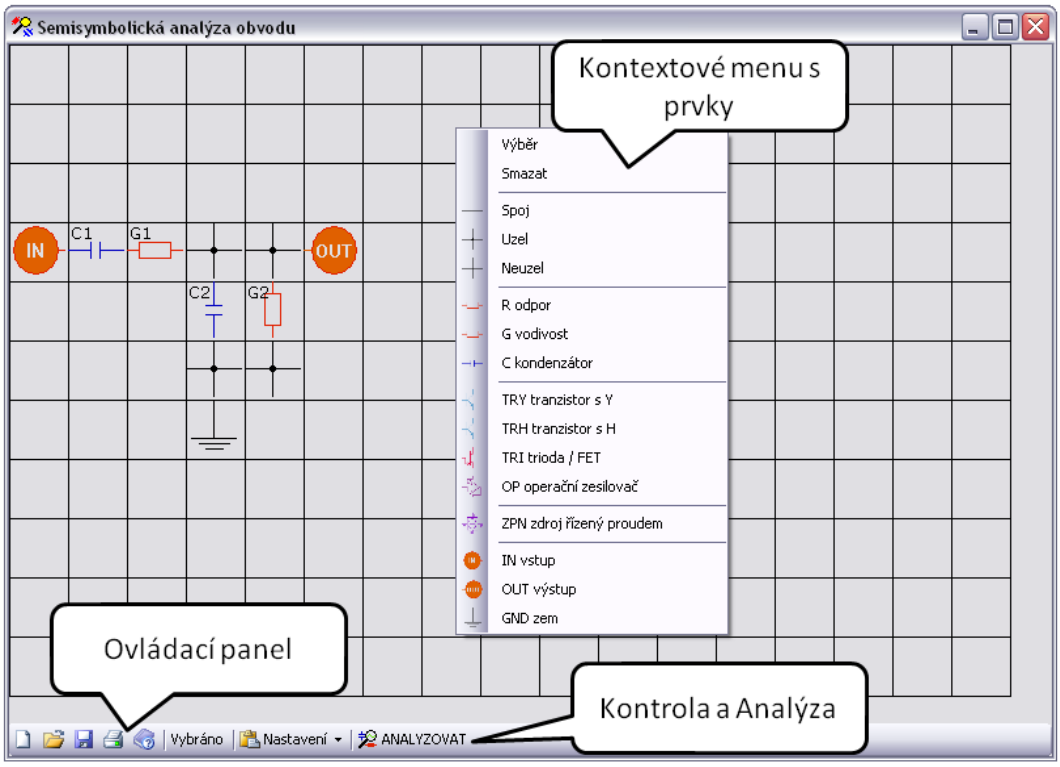
Výsledek je zobrazen na obrázku **Obr. 10**.



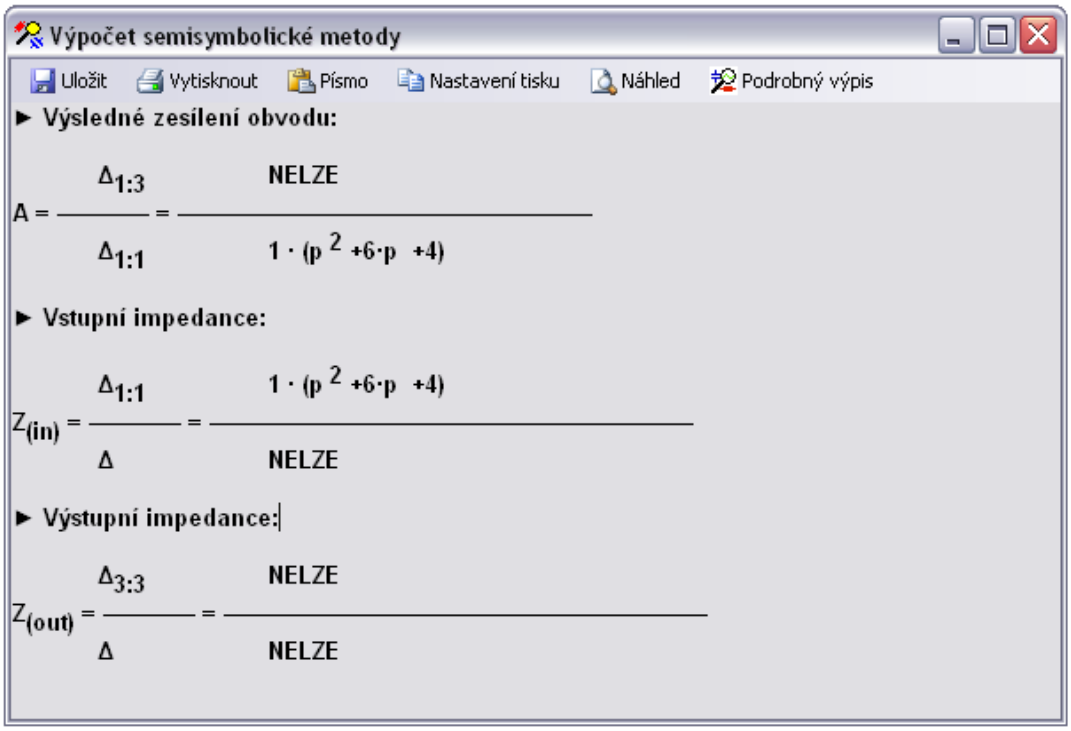
Obr. 10: Ukázka konzolové verze aplikace.

3.6 Finální aplikace

Výsledný program využívá technologii .NET Framework 3.5, která je vytvořena jako standardní formulářové okno s ovládacími prvky. Zadávání obvodu nyní není konzolové, ale přímo grafické. Uživatel po spuštění má před sebou okno, které má v sobě mřížku a ovládací panel. Pomocí pravého kliknutí myši v mřížce se zobrazí kontextové menu s prvky, kterými se zadá obvod. Levým tlačítkem myši se vybraný objekt vloží. Pokud je vybrán prvek, jenž má mít hodnotu, po zadání do mřížky se objeví jeho vlastnost, kde se zadává. Jestliže nevyhovuje natočení objektu, lze jej pomocí prostředního tlačítka myši otáčet. Po dokončení zadávání obvodu se klikne na tlačítko „ANALYZOVAT“, tím se provede kontrola na správnost zapojení a zobrazí se okno s výsledky, zde je možnost stručného zobrazení, nebo podrobného, ten ukáže postup výpočtu. V aplikaci je ovládací panel s mnoha funkcemi, například možnost uložení obvodu, otevření, nastavení viditelnosti mřížky atd.



Obr. 11: Ukázka programu, zadávací část.



Obr. 12: Ukázka výstupního okna s výsledky.

4 Popis kódu programu

4.1 Maticové funkce

4.1.1 Determinant

Nejdůležitější funkce pro výpočet je determinant matice, jelikož nevíme, jaký rozměr matice bude mít, je tedy řešen rekurzivním způsobem a matice může být n×n, bohužel determinant šel aplikovat jen na matice 3×3 a více, proto bylo nutné přidat ošetření na matice 1×1 a 2×2.[8]

```
//vypocet determinantu pro matici n*n rekurzivne
public double det(double[,] matice,int n)
{
    int radek = matice.GetLength(0); //zjisti pocet radku
    int h;
    double determinant = 0; //pro postupne pricitani
    double[,] mat = new double[radek, radek];

    if (n == 1) //determinant pro matici 1×1
    {
        determinant = matice[0, 0];
    }
    else if (n == 2) //determinant pro matici 2×2
        determinant = matice[0, 0]*matice[1, 1] - matice[1, 0]*matice[0, 1];
    else //detertminant n>2 × n>2
    {
        determinant = 0;
        for (int k = 0; k < n; k++)
        {
            for (int i = 1; i < n; i++)
            {
                h = 0;
                for (int j = 0; j < n; j++)
                {
                    if (j == k)
                        continue;
                    mat[i - 1, h] = matice[i, j];
                    h++;
                }
            }
            //vlastni funkce na pricitani subdeterminantu rekurzi
            determinant += Math.Pow(-1.0, k)*matice[0, k] * det(mat, n - 1);
        }
    }
    return determinant; //vrati vysledny determinant
}
```

4.1.2 Adjungovaná matice

Používá se pro vytvoření inverzní matice, inverzi řeší pomocí determinantů subdeterminantů. Matice A^{-1} má prvky $a_{i,j} = \frac{(-1)^{i+j}|A_{j,i}|}{|A|}$, kde $|A_{j,i}|$ je subdeterminant získaný z matice A vynecháním j-tého řádku a i-tého sloupce, |A| je determinant matice A. Funkce *adj(double[,] pole, int x, int y)*, slouží pro výpočet subdeterminantu matice A, tím že vynechá *x-řádek* a *y-sloupec*. Druhá funkce *adjungovana(double[,] pole)* spočítá čitatel zlomku.[7]

```
//pomocna funkce pro vypocet adjungovane matice
private double adj(double[,] pole, int x, int y)
{
    int radek = pole.GetLength(0);  //zjisti pocet radku
    int sloupec = pole.GetLength(1);    //zjisti pocet sloupcu
    int a = 0;
    int b = 0;
    //vytvori pomocnou matici ktera je mensi o jedno
    double[,] mat = new double[radek - 1, sloupec - 1];
    for (int i = 0; i < radek; i++)
    {
        for (int j = 0; j < sloupec; j++)
        {
            if (i == x) //vynecha radek
                continue;
            if (j == y) //vynecha sloupec
                continue;
            {
                //prenda prvky do pomocne aby vznikla zredukovana matice
                mat[a, b] = pole[i, j];
                b++;
                if (b >= (sloupec - 1))
                {
                    a++;
                    b = 0;
                }
            }
        }
    }
    return det(mat, radek - 1); //vypocita determinant zredukovane matice
}

//vypocet adjungovane matice
private double[,] adjungovana(double[,] pole)
{
    int radek = pole.GetLength(0); //zjisti pocet radku
    int sloupec = pole.GetLength(1); //zjisti pocet sloupcu
    double[,] mat = new double[radek, sloupec];
    for (int i = 0; i < radek; i++)
    {
        for (int j = 0; j < sloupec; j++)
        {
```

```
            //vypocita prvek adjung. matice dle vzorecku a je treba pricist
subdeterminant
            mat[j, i]=(Math.Pow(-1,((i + 1)+ (j + 1))))*this.adj(pole, i, j);
        }
    }
    return mat; //vrati adjungovanou matici
}
```

4.1.3 Inverzní matice

Výpočet inverzní matice je popsán výše u adjungované matice, inverzní funkce dělí jednotlivé prvky adjungované matice determinantem původní, tím se získají prvky inverzní matice. V kódu se nachází funkce *copy(pole,vysledek)*, která se používá na kopírování z jednoho pole do druhého.[7]

```
//inverse pro danou matici n×n pomoci adjungovane matice
public double[,] inverse(double[,] pole)
{
    int radek = pole.GetLength(0); //zjisti pocet radku
    int sloupec = pole.GetLength(1); //zjisti pocet sloupcu
    double[,] matAD = new double[radek, sloupec];
    matAD = this.adjungovana(pole); //vytvori adjungovanou matici k zadane
    //vypocita determinant k zadane matici
    double determinant = this.det(pole, radek);
    for (int i = 0; i < radek; i++)
    {
        for (int j = 0; j < sloupec; j++)
        {
            //vypocita jednotlive prvky inverzni matice
            pole[i, j] = matAD[i, j] / determinant;
        }
    }

    double[,] vysledek = new double[radek, sloupec];
    copy(pole, vysledek); //presune do vysledne matice

    return vysledek; //vrati vyslednou inverzni matici
}
```

4.2 Výpočetní jádro

4.2.1 Výpočet

Ukázka z výpočetní funkce, která má za parametry matice G a B, provede výpočet všech binominálních determinantů. Tyto výsledky jsou uloženy do *public proměnných*, ke kterým přistupuje poté aplikace, upraví znaménka a zobrazí výsledky, které byly spočítány. Kromě přímých výsledků jsou zde i rozpisy v nichž jsou uloženy postupné

úpravy matic. Postup je následující, vezme původní matice, předá je do funkce *uprava(matice, x, y)*, což vytvoří novou se zmenšeným rozměrem tak, že vynechá *x-ředek* a *y-sloupec*. Než dojde k samotnému výpočtu, zkontroluje, zda determinant subscentanční matice nevyjde nulový. Pokud je nenulový provede inverzní funkci. Dále součin $B^{-1} \cdot G$ ze kterých vznikne X, která se předá do Danilevského funkce (ukázka níže).

```
//samotny vypocet kde vstup je matice G a B ukazka pro vypocet delta11
public void vypocet(double[,] G, double[,] B)
{
    int n = B.GetLength(0); //zjisit velikost
    int n1 = n - 1; //zmensi velikost
    G1 = new double[n1, n1];
    B1 = new double[n1, n1];
    Y1 = new double[n1, n1];

    //prekopiruje aby nedoslo k upravam originalu
    G1 = fce.copy(this.uprava(G, 0, 0),G1);
    B1 = fce.copy(this.uprava(B, 0, 0),B1);

    B1I = new double[n1, n1];
    SB1IG1 = new double[n1, n1];
    RozpisD1 = new double[n1, n1, n1];
    RozpisT1 = new double[n1, n1, n1];
    RozpisT11 = new double[n1, n1, n1];
    //vynuluje rozpisy
    Array.Clear(RozpisD1, 0, RozpisD1.Length);
    Array.Clear(RozpisT1, 0, RozpisT1.Length);
    Array.Clear(RozpisT11, 0, RozpisT11.Length);

    //Y1 = G1 + p*B1
    C1 = fce.det(B1, n1); //kontrola zda lze vypocitat
    if (C1 != 0)
    {
        typ = 0;//urci deltu
        //prekopiruje aby nedoslo k upravam originalu
        B1I = fce.copy(B1, B1I);
        B1I = fce.inverse(B1I); //provede inverzi
        //soucin B1*G pro zjisteni X
        SB1IG1 = fce.copy(fce.soucin(B1I, G1), SB1IG1);
        //danilevskeho metoda pro upravu T1*X*T
        Y1 = fce.copy(danilevsky(SB1IG1), Y1);
        //vezme prvni radek z upravene matice jsou zde koeficienty
        Delta11 = delta(Y1);
    }
```

4.2.2 Úprava matic

Funkce ze zadané matice vytvoří novou, ve které se nebudou nacházet *x-řádek* a *y-sloupec*. Poté provede určení znaménka, roznásobením jednotlivých prvků $(-1)^{x+y} \cdot a_{i,j}$.

```
//vynecha v matici X-ty radek a Y-ty sloupec a vrati jako nove pole
private double[,] uprava(double[,] pole, int x,int y)
{
    int rozmer = pole.GetLength(0); //zjisti velikost matice
    int a = 0;
    int b = 0;

    double[,] mat = new double[rozmer-1, rozmer-1];
    for (int i = 0; i < rozmer; i++)
    {
        for (int j = 0; j < rozmer; j++)
        {
            if (i == x)
                continue; //vynecha radek
            if (j == y)
                continue; //vynecha sloupec
            {
                //prenda prvky do pomocne aby vznikla zredukovana matice
                mat[a, b] = pole[i, j];
                b++;
                if (b >= (rozmer - 1))
                {
                    a++;
                    b = 0;
                }
            }
        }
    }
    //znamenko upravi dle upravy delta 1:n
    for (int i = 0; i < rozmer-1; i++)
    {
        for (int j = 0; j < rozmer-1; j++)
        {
            //vynasobi -1 na x+y prvky upravene matice
            mat[i,j]=Math.Pow(-1, ((x + 1) + (y + 1))) * mat[i, j];
        }
    }

    return mat; //vrati novou matici
}
```

4.2.3 Danilevského metoda

Je nejdůležitější funkce celého výpočtu, upravuje postupně matici X dle podobnostní transformace $T^{-1} \cdot X \cdot T$ dokud nedosáhne Frobeninova tvaru. Postup je následující, vytvoří si pomocná dvourozměrná pole, jedno má na diagonále jedničky a další je má až od druhého řádku. Jedničkové pole se využívá při vytváření T^{-1} a z X se do ní vloží řádek. Poté proběhne součin matic, provede se kontrola, zda tvar X matice bez prvního řádku odpovídá tvaru matice pomocné a jedničkami na diagonále od druhého řádku.

```
//uprava matice X pomoci danilevskeho metody T*X*T
public double[,] danilevsky(double[,] X)
{
    bool test; //kontrola podobnosti
    int rozmer = X.GetLength(0); //zjisti rozmer matice
    double[,] mat = new double[rozmer, rozmer]; //pomocna
    double[,] TX = new double[rozmer, rozmer]; //soucin T1 * X
    double[,] T1 = new double[rozmer, rozmer]; //vytvorena T1 matice
    double[,] T = new double[rozmer, rozmer]; //inverzni matice k T
    //matice D s 1 na diagonale zbytek 0
    double[,] D = new double[rozmer, rozmer];
    //Xx matice pomocna k porovnani zda splnuje pozadovany tvar
    double[,] Xx = new double[rozmer, rozmer];
    double[,] vysledek = new double[rozmer, rozmer]; //vysledna matice

    for (int i = 0; i < rozmer; i++)
    {
        for (int j = 0; j < rozmer; j++)
        {
            //vytvori maitici jenz ma od 2. radku na diagonale 1 zbytek 0
            if ((i == j) && (j - 1 >= 0))
            {
                Xx[i, j - 1] = 1;
            }
            else
            {
                Xx[i, j] = 0;
            }
        }
    }

    for (int i = 0; i < rozmer; i++)
    {
        for (int j = 0; j < rozmer; j++)
        {
            //matice s 1 na diagonale zybtek 0
            if (i == j)
            {
                D[i, j] = 1;
            }
            else
            {
                D[i, j] = 0;
            }
        }
    }

    test=false;
    int k = rozmer - 1;
    mat = fce.copy(X, mat); //presune X do mat
    int z=0; //pro pomocny vypis

    do //cyklus ktery upravuje dokud matice nejsou stejne jako predloha Xx
    {
        rozpis(mat, z); //zapise upravu X
        T1 = fce.copy(D, T1); //zkopiruje prvky z D do T1, na diagonále 1ky

        for (int i = 0; i < rozmer; i++)
        {
            //tvori T1 matici tim ze vezme k-ty radek z X
            T1[k - 1, i] = mat[k, i];
        }
```

```
        T = fce.copy(T1, T); //zkopiruje prvky z T1 do T
        T = fce.inverse(T); //inverse matice T
        TX = fce.soucin(T1, mat); //soucin T1*X
        vysledek = fce.soucin(TX, T); //soucin TX*T => Xn
        vysledek = fce.copy(vysledek, mat); //preda do vysledku
        k--; //zmenusje radky pro upravu od zadu
        rozpisT(T, z); //zapise upravu T
        rozpisT1(T1, z); //zapise upravu T1
        z++; //zvetsi polozku jako dalsi krok
        test = kontrola(vysledek, Xx); //zkontroluje na podobnost

        if (k <= 0) //porovnava zda neprekrocil pocet radku
            test = true;
    } while (test != true);

    return vysledek; //vrati vyslednou matici
}
```

4.3 Vlastnosti prvků

Prvek se vkládá do matice jako číslo ID, to se definuje jako třída, která obsahuje více vlastností, například hodnoty prvků, pozice, rotace, typ, obsahuje dále funkci *Clear()* která, defaultně nastaví vlastnosti.

```
public class Objekt
{
    public int id; //identifikator prvku
    public double[] value; //hodnoty prvku
    public string description; //popisek prvku
    public int rotate; //natoceni prvku
    public int type; //typ prvku R,C,G...
    public int[] uzel; //uzly spojeni
    public Point[] ruld; //Pointy X,Y spojeni v ose

    public Objekt()
    {
        Clear(); //konstruktor vynuluje dany objekt
    }
    //nastavi defaultni hodnoty prvku
    public void Clear()
    {
        id = 0;
        value = new double[4];
        description = string.Empty;
        rotate = 0;
        type = 0;
        uzel = new int[4];
        ruld = new Point[4];
        Array.Clear(ruld, 0, ruld.Length); //vynuluje pole
        Array.Clear(value, 0, value.Length); //vynuluje pole
        Array.Clear(uzel, 0, uzel.Length); //vynuluje pole
```

```
        for (int i = 0; i < 4; i++)
        {
            //nastavi defaultni hodnoty na -1 jelikoz 0 se pouziva normalne
            ruld[i].X = -1;
            ruld[i].Y = -1;
            uzel[i] = -1;
        }
    }
}
```

4.4 Aplikace

4.4.1 Vykreslení

Slouží pro vykreslení prvků, mřížky i popisků do Formuláře, vykresluje se do pomocné bitmapy a ta se poté zavolá v události *Paint()*.Typy prvků jsou načteny z vlastností prvků i s rotací, pomocí typu se vloží do dané oblasti odpovídající obrázek s otočením.[9]

```
//vykresleni
public Bitmap kreslit()
{
    //nastaveni rozmeru bitmapy
    vyska = (this.Height - toolStrip1.Height) / rozmer;
    vyska--;
    radek = (this.Width) / rozmer;

    Bitmap vysledek = new Bitmap(this.Width, this.Height);

    //vykresli mrizku
    if (mrizka == true)
    {
        //vykresli do vysledne bitmapy
        Graphics gfx = Graphics.FromImage(vysledek);
        //styl vykresleni
        gfx.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighSpeed;

        Pen myPen = new Pen(Color.Black); //nastaveni pera
        for (int i = 0; i < this.Height; i = i + rozmer)
        {
            for (int j = 0; j < this.Width; j = j + rozmer)
            {
                //vykresleni
                gfx.DrawLine(myPen, 0, i, radek * rozmer, i);
                gfx.DrawLine(myPen, j, 0, j, vyska * rozmer);
            }
        }
        gfx.Dispose(); //uvolni
    }
    Point pozice; //pozice prvku z mapy do mrizky
    //vykresli do vysledne bitmapy
    Graphics grafika = Graphics.FromImage(vysledek);
```

```
    //styl vykresleni
    grafika.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighSpeed;

    Bitmap symbol;
    for (int i = 0; i < pocet; i++)
    {
        for (int j = 0; j < pocet; j++)
        {
            if (mapa[i, j] != 0)
            {
                //vyhleda prvke v mape a vykresli do daneho mista
                pozice = hledat(mapa[i, j]);
                symbol = image_prvek(prvky[pozice.X, pozice.Y].type);
                for (int l = 0; l < 4; l++)
                {
                    if (prvky[pozice.X, pozice.Y].rotate == rotace[l])
                        symbol = rotator(symbol, l);//nastavi natoceni prvku
                }
                //vykresli prvek na dsane souradnici
                grafika.DrawImage(symbol, pozice.Y * rozmer,pozice.X *rozmer,
                symbol.Width, symbol.Height);
                //vykresli popisek daneho prvku
                if ((popisky == true) && (mapa[i, j]>5000))
                    //funkce na vykresleni textu
                    grafika.DrawString(prvky[pozice.X, pozice.Y].description,
                    new Font("Verdana", 10), new SolidBrush(Color.Black),
                    (pozice.Y * rozmer), (pozice.X * rozmer ));
                symbol.Dispose(); //uvolni
            }
        }
    }
    grafika.Dispose(); //uvolni
    return vysledek; //vrati vyslednou bitmapu
}
```

4.4.2 Výběr, vkládání a vymazávání prvků

Sestavování obvodu se dělá pomocí myši, kliknutím na mřížku, pravé tlačítko myši zobrazí kontextové menu s prvky, spoji, výběrem a mazáním. Pokud myš není v oblasti mřížky, nic se neprovede. Po výběru prvku z menu se změní kurzor na obrázek prvku, prostředním tlačítkem myši je možné nastavovat jeho rotaci, levým tlačítkem se prvek vloží pokud, na daném místě není objekt. Každý prvek i spoje mají vlastní ID číslo, kromě tří základních IN, OUT, GND, ty se řadí mezi speciální prvky. Toto ID se zapíše do mapy a tím je určena jeho poloha v mřížce pro vykreslení.

```
//klinuti mysi
private void Form1_MouseClick(object sender, MouseEventArgs e)
{
    //zjisti na jakem ctverecku se ukazatel nachazi
    int y = e.X / 45;
    int x = e.Y / 45;
```

```
    //pokud je na mape a ne mimo reaguje na dalsi funkce
    if((x <vyska)&&(y<radek))
    {
        //leve tlacitko mysi s vyberem mazani, smaze dany prvek na pozici X,Y
        if ((e.Button == MouseButtons.Left) && (select < 0))
        {
            //je vybrano mazani prvku
            if (mapa[x, y] > 0)
            {
                //pokud se jedna o speciální prvek provede inkrementaci
                //poctu aby bylo mozne ho znovu vlozit
                if ((mapa[x, y]==10) || (mapa[x, y]==11) || (mapa[x, y]==12))
                    oznaceni[mapa[x,y]]--;
                prvky[x, y].Clear();
                mapa[x, y] = 0;
            }
            else
            {
                mapa[x, y] = 0;
            }
            Invalidate(); //prekresleni
        }
        //leve tlacitko mysi s vyberem krom mazani provede vlozeni
        //prvku na dane misto X,Y
        else if ((e.Button == MouseButtons.Left)&&(select>-1))
        {
            //pokud neni na danem miste objekt vytvori novy a rozlisi
            //zda neni specialni
            if ((mapa[x, y] == 0)&&(select>0)&&(select!=10)&&(select!=11)&&
            (select!=12))
            {
                int id;
                if(select>11)
                    //objekty v tomto rozmezi urcuji spoje, uzly a neuzly
                    id = generator_id(100,5000);
                else
                    id = generator_id(5001,32000); //ozanucji prvky
                mapa[x, y] = id;
                prvky[x, y].id = id;
                prvky[x, y].description = prvky_name[select] +
                (++oznaceni[select]);
                prvky[x, y].type = select;
                mezisouradnice(x, y, select, rot);
                //pokud jde o prvek vyvoval okno s vlastnostmi
                if (id >= 5001)
                    vlastnost(id);
                else
                    Invalidate(); //prekresli
            }
            //pokud jde o specialni objekt jeho vlozeni se provede
            //funkci spec_prvek jenz hlida vlozeni jen jednou
            else if ((mapa[x, y] == 0) && (select>9) && (select<13))
            {
                if (oznaceni[select]<1)
                {
                    spec_prvky(x, y, select);
                }
            }
```

```
            //pokud na miste prvke je otevre okno s vlastnostmi
            else if(mapa[x,y]>=5001)
            {
                vlastnost(mapa[x,y]);
            }
        }
        //prostredni tlacitko mysi s vyberem objektu umozni natoceni
        else if ((e.Button == MouseButtons.Middle)&&(select>0)&&(select!=9))
        {
            rot++;
            if (rot > 3)
            {
                rot = 0;
            }

            Bitmap b;
            b = this.image_prvek(select); //vybere obrazek objektu

            b = rotator(b, rot); //natoci obrazku dle kliknuti

            ukazatel_mysi(b); //nastavi ukazatel mysi uz s natocenim obrazku

            b.Dispose(); //prekresleni
        }
        else // prave tlacitko mysi vyvola kontextove menu od pozice X,Y
        {
            Point point1 = Cursor.Position;
            this.contextMenuStrip1.Visible = true;
            this.contextMenuStrip1.Show(point1);
        }
    }
}
```

4.4.3 Sestavení matic

Aby bylo možné vytvořit z daného obvodu impedanční matici, je nutné, aby došlo k přiřazení uzlů k prvkům, to probíhá následujícím postupem:

- Přiřazení spojnic vývodům všech objektů.
- Redukcí spojů, uzlů a neuzlů, tak že se nahradí jedním bodem, pokud tedy G a C je spojen Spojem, dojde k jeho odstranění a nahrazením jedním bodem který bude společný u G a C.
- Zredukování nezapojených prvků a odstranění duplicitních spojnic.
- Přiřazení spojnicím čísla uzlů.

Po přiřazení uzlů je možné sestavit matice G a B, které se předají poté do výpočetního jádra. Tato funkce vezme prvek, zjistí typ, natočení, uzly, poté provede kontrolu na zkratování a vloží na dané uzly hodnotu, pokud jde o trojpól dojde předem

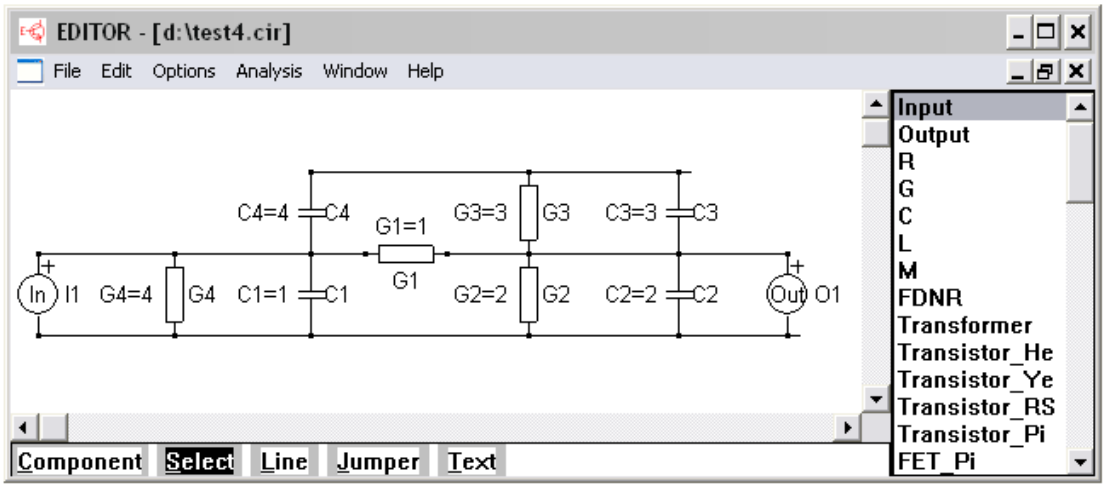
k přečíslování pozic uzlů. Ukázka kódu ukazuje sestavení matice tranzistoru s Y-parametry, podobně se to provádí u ostatních prvků, každý z nich má, ale jiné rovnice.

```
//sestaveni matic G a B dosazenim hodnot prvku
private void matice_sestava(int x,int y,int typ)
{
    bool nulak = false;
    int[] pole= new int[4];
    int poc=0;
    int n = matice_rozmer;
    for (int i = 0; i < 4; i++)
    {
        if (stav[x, y].uzel[i] > -1)
        {
            pole[poc++]=stav[x, y].uzel[i];
        }
    }

    int uzel1=-1;  //B
    int uzel2=-1;  //C
    int uzel3=-1;  //E

    //u trjpolu zalezi na natoceni jelikoz se meni vstupy a vystupy,
    //dochazi proto k precislovani uzlu
    if ((stav[x, y].type > 4) && (stav[x, y].type < 9))
    {

        if (stav[x, y].rotate == 90)
        {
            uzel1 = stav[x, y].uzel[3]; //B
            uzel2 = stav[x, y].uzel[2]; //C
            uzel3 = stav[x, y].uzel[0]; //E
        }
        else if (stav[x, y].rotate == 180)
        {
            uzel1 = stav[x, y].uzel[0]; //B
            uzel2 = stav[x, y].uzel[3]; //C
            uzel3 = stav[x, y].uzel[1]; //E
        }
        else if (stav[x, y].rotate == 270)
        {
            uzel1 = stav[x, y].uzel[1]; //B
            uzel2 = stav[x, y].uzel[0]; //C
            uzel3 = stav[x, y].uzel[2]; //E
        }
        else
        {
            uzel1 = stav[x, y].uzel[2];    //B
            uzel2 = stav[x, y].uzel[1];    //C
            uzel3 = stav[x, y].uzel[3];    //E
        }
    }
    //kazdy prvek ma sve rovnice, ktere se zapisi do matice, musi se davat
    //pozor na to zda neni spojen se zemi, pokud ano tak matice je jinaci
    switch (typ)
    {
        case 5:
            {
```

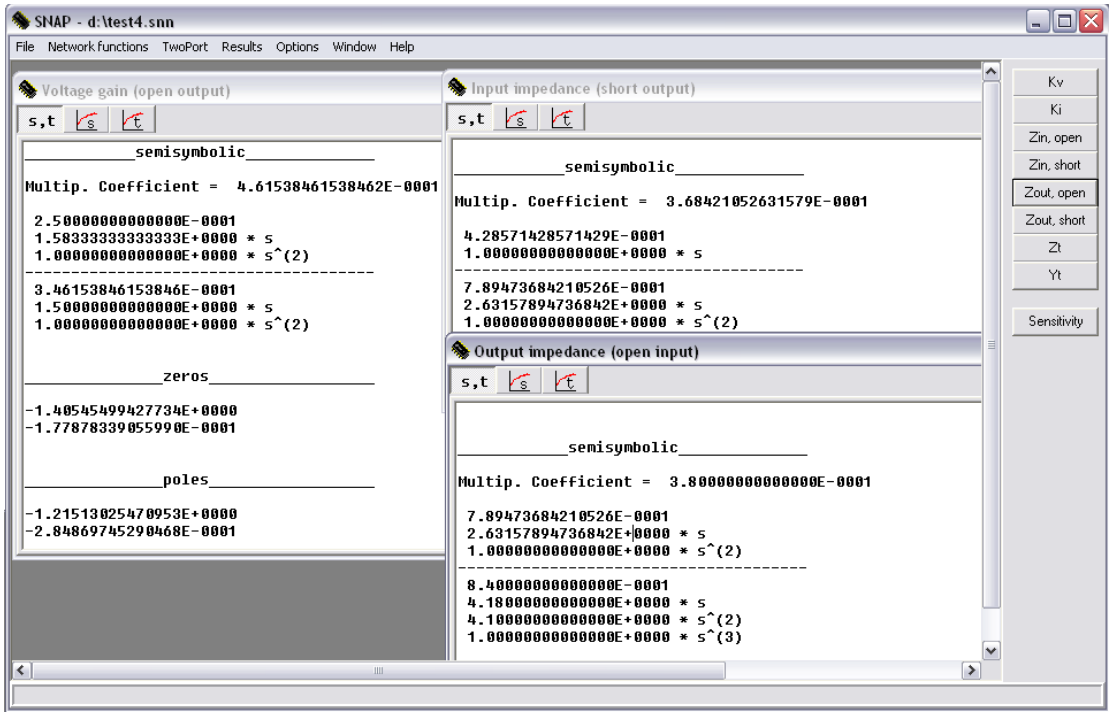
```
                //kontrola zda neni zkrat
                if ((uzel1 == uzel2) || (uzel1 == uzel3) || (uzel2 == uzel3))
                {
                    MessageBox.Show("Tranzistor " + stav[x, y].description +
                    " je
                    zkratován!", "CHYBA PRVKU", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                    break;
                }
                double[] hodnota = new double[4];
                hodnota=stav[x, y].value;
                if (spojen_se_zemi(uzel1)==true) //B na zem
                {
                    G[uzel2, uzel2] = G[uzel2, uzel2] + hodnota[3];
                    G[uzel2, uzel3] = G[uzel2, uzel3] + (-hodnota[2] –
                    hodnota[3]);
                    G[uzel3, uzel2] = G[uzel3, uzel2] + (-hodnota[1] –
                    hodnota[3]);
                    G[uzel3, uzel3] = G[uzel3, uzel3] + (hodnota[0] +
                    hodnota[1] + hodnota[2] + hodnota[3]);
                }
                else if(spojen_se_zemi(uzel2)==true) //C na zem
                {
                    G[uzel1, uzel1] = G[uzel1, uzel1] + hodnota[0];
                    G[uzel1, uzel3] = G[uzel1, uzel3] + (-hodnota[0] –
                    hodnota[1]);
                    G[uzel3, uzel1] = G[uzel3, uzel1] + (-hodnota[0] –
                    hodnota[2]);
                    G[uzel3, uzel3] = G[uzel3, uzel3] + (hodnota[0] +
                    hodnota[1] + hodnota[2] + hodnota[3]);
                }
                else if(spojen_se_zemi(uzel3)==true) //E na zem
                {
                        G[uzel1, uzel1] = G[uzel1, uzel1] + hodnota[0];
                        G[uzel1, uzel2] = G[uzel1, uzel2] + hodnota[1];
                        G[uzel2, uzel1] = G[uzel2, uzel1] + hodnota[2];
                        G[uzel2, uzel2] = G[uzel2, uzel2] + hodnota[3];
                }
                else
                {
                    G[uzel1, uzel1] = G[uzel1, uzel1] + hodnota[0];
                    G[uzel1, uzel2] = G[uzel1, uzel2] + hodnota[1];
                    G[uzel1, uzel3] = G[uzel1, uzel3] + (-hodnota[0] –
                    hodnota[1]);
                    G[uzel2, uzel1] = G[uzel2, uzel1] + hodnota[2];
                    G[uzel2, uzel2] = G[uzel2, uzel2] + hodnota[3];
                    G[uzel2, uzel3] = G[uzel2, uzel3] + (-hodnota[2] –
                    hodnota[3]);
                    G[uzel3, uzel1] = G[uzel3, uzel1] + (-hodnota[0] –
                    hodnota[2]);
                    G[uzel3, uzel2] = G[uzel3, uzel2] + (-hodnota[1] –
                    hodnota[3]);
                    G[uzel3, uzel3] = G[uzel3, uzel3] + (hodnota[0] +
                    hodnota[1] + hodnota[2] + hodnota[3]);
                }
                break;
            }
    }
}
```

5 Testování programu

Ověření výsledku ukázkového příkladu mé aplikace s výsledkem řešeným v programu SNAP. Jako první je obvod vytvořený v Editoru SNAP, jsou zde zobrazeny i hodnoty prvků a posléze výsledek analýzy. SNAP má parametr $s = j \cdot \omega$ v mé aplikace je parametr p.

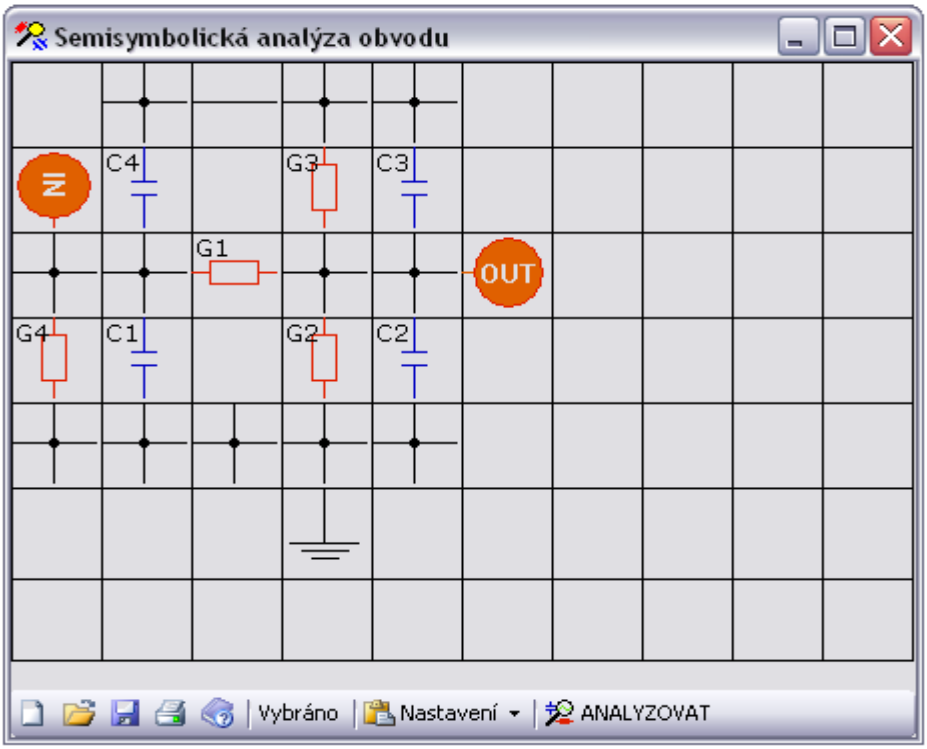


Obr. 13: Obvod navržený v Editoru programu SNAP.

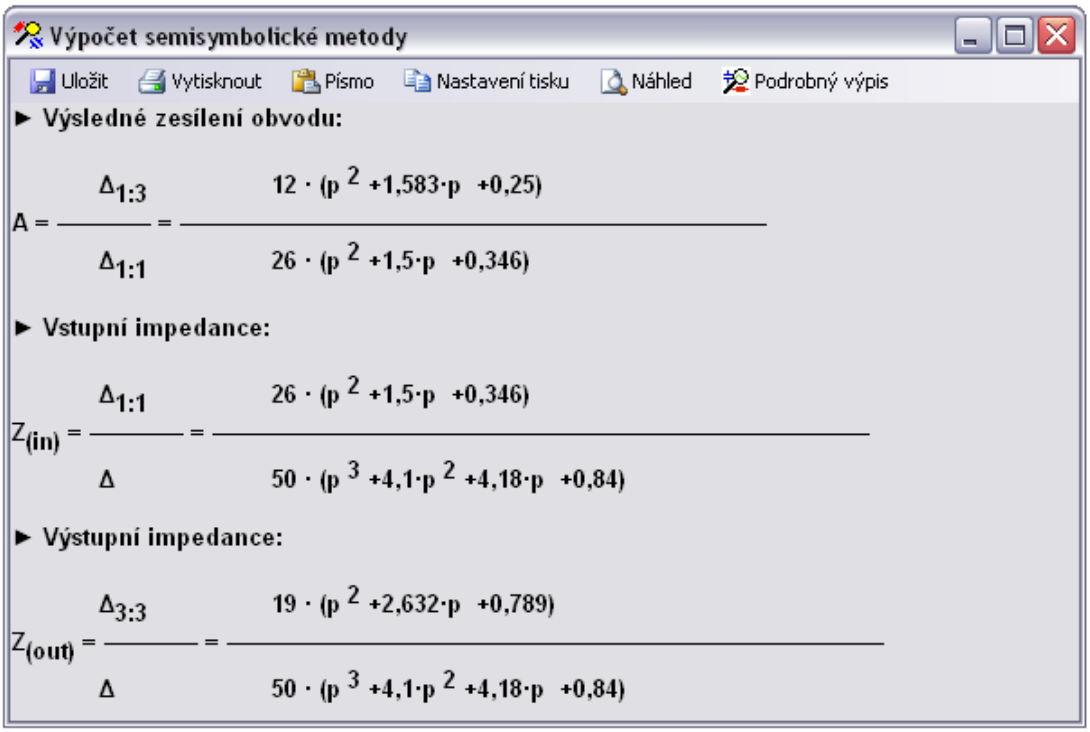


Obr. 14: Výsledek napěťového přenosu, vstupní a výstupní impedance v programu SNAP.

Zde je zobrazen stejný obvod jako na **Obr. 13** a k němu výsledek. Hodnoty prvků jsou též stejné, bohužel má aplikace je nezobrazuje, jen popisky. Po porovnání **Obr. 14** a **Obr. 16** zjistíme, že výsledky jsou totožné.



Obr. 15: Obvod navržený v mé aplikaci.



Obr. 16: Výsledek napěťového přenosu, vstupní a výstupní impedance v mé aplikaci.

6 Srovnání s obdobnými aplikacemi

Existují podobné programy komerční i nekomerční, které řeší výpočty přenosu napětí. Provedu srovnání s českými programy SESYCO a SNAP.

6.1 SESYCO program

Jedná se o vývojový nástroj pro návrh analogových obvodů, využívající semisymbolický numerický algoritmus. Pracuje na operačním systému MS-DOS a byl naprogramován v jazyku Pascal tak, aby měl co nejmenší nároky na rychlost počítače i operační paměti. Obvod se zadává dialogovým způsobem, například takto:

Příkaz [*parametr*] [*parametr*] … [*parametr*]

SESYCO nalezne přenosovou funkci ve tvaru dvou mnohočlenů a z ní počítá zisk a skupinové dění v zadaném kmitočtovém intervalu. V některých verzích umí aplikace zobrazit pás rozptylu charakteristik v závislosti na tolerancích součástek, vypočítat nuly a póly přenosových funkcí. Program trpí nevýhodou semisymbolického algoritmu, že je lze použít jen na malé množství obvodů. Rozdíly této a mé aplikace jsou zřejmé, výpočetní jádro pracuje stejně, počítání se provádí přes semisymbolický numerický algoritmus, ale liší se metoda zadávání, zde je pomocí konzole a u mé je grafické navrhování obvodu. Dalším rozdílem jsou prvky, které lze zadat, v SESYCu je například ideální transformátor.[1]

6.2 SNAP program

Píši zde drobné informace o tom to programu, jelikož sloužil pro ověřování výsledků z mé aplikace. SNAP neřeší přímo semisymbolickou analýzu, ale počítá symbolickou analýzu pokud jsou přidány hodnoty prvků, dosadí do symbolického výsledku a získá tak semisymbolický, poté může sestavit charakteristiky. Jedná se o aplikaci, která má hodně možností, skládá se z Editoru, kde se sestavuje obvod a výpočetní části, ve které jsou zobrazeny výsledky. Rozdíl oproti mému programu je takový, že můj algoritmus řeší přímo semisymbolickou analýzu bez nutnosti symbolického výsledku, proto je rychlejší výpočet u složitějších obvodů. Dále SNAP Editor má vlastní sytém spojování prvků pomocí čar, u mého programu je toho nahrazeno spoji či uzly. Díky systému vkládání do mřížek nedochází k překrývání prvků jako u SNAPu.[2]

7 Závěr

Záměrem mé práce bylo vytvořit aplikaci, která umožní jednoduše nakreslit schéma, poté ho zkontrolovat na správnost zapojení, dále vypočítat pomocí semisymbolického numerického algoritmu a zobrazit výsledný napěťový přenos, vstupní a výstupní impedanci ve formě rovnic s neznámou p, jenž je Laplacovým operátorem. Po naprogramování jsem otestoval funkčnosti této metody a zjistil její nevýhodu, kterou je použití omezeno na třídu ne příliš rozsáhlých obvodů. Program splňuje zadání ve všech bodech, jedná se o grafickou aplikaci vytvořenou pro operační systém Windows v jazyce C#, která využívá technologii .NET Framework 3.5. Obsahuje základní kontrolu obvodu, jako je například vložení klíčových prvků (IN, OUT, GND), kontrolu zkratu a nezapojených prvků. Nevýhodou je, že chybí ošetření na vytvoření uzavřeného obvodu bez spojení s hlavním schématem. Dále zde chybí jeden ze základních prvků a to cívka, důvod je prostý, nelze ji přičítat do matice imaginárních složek, jelikož se při výpočtech touto metodou parametr p ($j \cdot \omega$) vytýká. Po rozšíření susceptance pro cívku $\frac{1}{L \cdot p}$ by zbylo ve jmenovateli $\omega^2 \cdot L$ a v mém výpočtu se ω nezadává. O tuto schopnost by má aplikace mohla být rozšířena a nato mohla zobrazit výsledný graf. Program je z pohledu programového omezen ve velikosti počtu prvku a ten je stanoven na 50×50 objektů. Také by mohl někdo vytknout velikost pracovní plochy, ta neumožňuje scrollování. Aplikace využívá vždy aktuální velikosti okna, tedy pokud chceme větší, musíme maximalizovat. Program se skládá z několika částí a jednou z nich je výpočetní jádro, bohužel jsem ho navrhnul tak, že využívá matici vytvořenou z nejméně tří uzlů. Na tuto chybu v prvotní verzi dopláceli jednoduché obvody o počtu dvou uzlů typu RC článek a podobně. Naštěstí se mi podařilo tento problém ve finální verzi vyřešit tak ,že výpočet je řešen mimo jádro přímo v části výstupu. Protože je zadání vytvořeno pro účely výuky, uznal jsem za vhodné, aby byl vidět podrobný postup řešení analýzy a vytvořil tak ve výstupním okně možnost zobrazit jak stručný výsledek, tak i podrobný postup řešení.

Seznam použité literatury

BIOLEK, D. Navrhování elektronických obvodů počítačem. Skripta UO Brno, 2004, 260s. Dostupné z WWW: <http://user.unob.cz/biolek/vyukaVA/skripta/NOP.pdf>.

BIOLEK, D. Analogové elektronické obvody. Laboratorní cvičení. Elektronické učební texty, 41 s., ÚMEL FEKT VUT Brno, 2003.

BRTNÍK, B. Simulace elektronických obvodů. Skripta VSP Jihlava, 2010.

BRTNÍK, B. Elektrické obvody I. Skripta VSP Jihlava, 2007.

Http://msdn.microsoft.com/en-us/vcsharp [online]. 2011 [cit. 2011-05-22]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/vcsharp>.

Inverzní matice [online], poslední aktualizace 1. Května 2011 16:17,Wikipedie. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Inverzn%C3%AD_matice>

MAŤÁTKO, Jan. *Kniha: elektronika* . Vyd. 5. [s.l.] : IDEA SERVIS, 2002. 325 s. ISBN 80-85970-42-2.

MELICHAR, Roman. *Přímé metody výpočtu charakteristických čísel matic* [online]. Brno, 2007. 51 s. Bakalářská práce. Masarykova Univerzita v Brně. Dostupné z WWW: <http://www.math.muni.cz/~xmelicha/b20070524.pdf>.

PAVLÍK, Petr; SOBOTKA, Václav. SESYCO : vývojový nástroj pro návrh analogových obvodů. *SLABOPROUDÝ OBZOR 50 : PŘÍLOHA PRO MLADÉ INŽENÝRY.* 1989, 50, 9.

SHARP, John. *Kniha: Microsoft Visual C# 2010 -- Krok za krokem* . Vyd. 1. [s.l.] : Computer press, 2011. 696 s. ISBN 978-80-251-3147-3.

Seznam citace

[1] PAVLÍK, Petr; SOBOTKA, Václav. SESYCO : vývojový nástroj pro návrh analogových obvodů. *SLABOPROUDÝ OBZOR 50 : PŘÍLOHA PRO MLADÉ INŽENÝRY*. 1989, 50, 9, s. 10-15.

[2] BIOLEK, D. Navrhování elektronických obvodů počítačem. Skripta Brno, 2004, s. 6-10.

[3] BRTNÍK, B. Simulace elektronických obvodů. Skripta VSP Jihlava, 2010, s. 1-2, s. 30-33.

[4] C Sharp [online], poslední aktualizace 7. Května 2011 10:30 [cit. 16. 4. 2007], Wikipedie. Dostupné z WWW: <http://cs.wikipedia.org/wiki/C_Sharp>

[5] Lineární obvod : sdělovací technika. In *Leccos - lineární obvod* [online]. [s.l.] : [s.n.], 2009 [cit. 2011-05-24]. Dostupné z WWW: <http://leccos.com/index.php/clanky/linearni-obvod>.

[6] BRTNÍK, B. Elektrické obvody I. Skripta VSP Jihlava, 2007, s. 37-49.

[7] Inverzní matice [online], poslední aktualizace 1. Května 2011 16:17 [cit. 16. 4. 2007], Wikipedie. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Inverzn%C3%AD_matice>

[8] ALI G_3, Ali G_3. Determinant- Recursive Algorithm. *Dream.In.Code : Programming Help - C#* [online]. 29.3.2010, n. 1, [cit. 2011-05-24]. Dostupný z WWW: <http://www.dreamincode.net/forums/topic/164932-determinant-recursive-algorithm/>.

[9] NARAYANASWAMY, Anand . Graphics Programming Using C#. In Microsoft & .NET - Visual C# [online]. [s.l.] : [s.n.], 30.6.2002 [cit. 2011-05-24]. Dostupné z WWW: <http://www.developer.com/net/csharp/article.php/1435391/Graphics-Programming-Using-C.htm>.

Seznam obrázků