VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

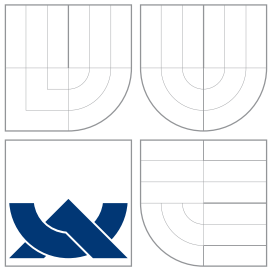
PARAZITNÍ INDUKČNOSTI PŘI ŘEŠENÍ ELEKTRICKÝCH OBVODŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

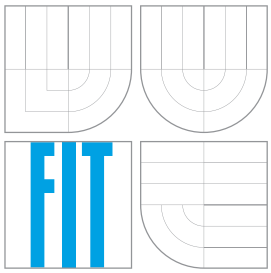
AUTOR PRÁCE
AUTHOR

ROMAN ŠEVČÍK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PARAZITNÍ INDUKČNOSTI PŘI ŘEŠENÍ ELEKTRICKÝCH OBVODŮ

ELECTRONIC CIRCUITS SIMULATIONS AND PARASITIC INDUCTANCES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN ŠEVČÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. JIŘÍ KUNOVSKÝ, Csc.

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Ševčík Roman**
Obor: Informační technologie
Téma: **Parazitní indukčnosti při řešení elektrických obvodů**
Kategorie: Teorie obvodů

Pokyny:

1. Seznamte se s problematikou numerického řešení obyčejných diferenciálních rovnic s přímým využitím Taylorovy řady a jejich řešením v TKSL.
2. Vypracujte metodiku řešení elektrických obvodů s parazitními indukčnostmi s využitím techniky diferenciálního počtu.
3. Proveďte analýzu stability numerického řešení, počtu rovnic a složitosti matematických úprav.
4. Výpočty ověřte v TKSL.
5. Srovnejte výsledky se světovými standardy.
6. Metodické postupy aplikujte při návrhu programového vybavení pro automatické generování soustavy ekvivalentních diferenciálních rovnic.

Literatura:

- Dle zadání vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kunovský Jiří, doc. Ing., CSc.**, UITS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Roman Ševčík**
Id studenta: 84075
Bytem: Nevšová 67, 763 21 Slavičín
Narozen: 11. 03. 1985, Zlín
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Parazitní indukčnosti při řešení elektrických obvodů
Vedoucí/školitel VŠKP: Kunovský Jiří, doc. Ing., CSc.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☒ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Tato práce se zabývá řešením diferenciálních rovnic a soustav diferenciálních rovnic pomocí použití parazitních kapacit. Takto získané rovnice slouží jako vstupní data pro simulační nástroj TKSL, který umožňuje jejich řešení pomocí Taylorovy řady. Dále je podle návrhu v jazyce C# implementován systém pro grafické zobrazení výstupních dat z TKSL. Systém umožňuje zobrazení časových průběhů jako dvourozměrné spojité grafy.

Klíčová slova

Grafický interaktivní systém, diferenciální rovnice, TKSL, parazitní cívky, graf, .NET

Abstract

This work deals with the solution of differential equations and systems of differential equations through the use of parasitic inductances. Thus obtained equations serve as input data for the simulation tool TKSL, which allows their solutions using the Taylor series. Furthermore, according to the proposal in C# is implemented a system for graphic display output data from TKSL. The system makes it possible to view trending as a continuous two-dimensional graphs.

Keywords

Graphic interactive system, differentials equations, TKSL, parasitic inductances, graph, .NET

Citace

Roman Ševčík: Parazitní indukčnosti při řešení elektrických obvodů, bakalářská práce, Brno, FIT VUT v Brně, 2008

Parazitní indukčnosti při řešení elektrických obvodů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Jiřího Kunovského Csc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Ševčík
10.května 2008

Poděkování

Děkuji Doc. Jiřímu Kunovskému za jeho odbornou pomoc a cenné rady při vzniku této práce a také všem otatním, kteří mi jakkoliv s prací pomohli.

Obsah

Kapitola 1

Úvod

Způsobů jak řešit elektrické obvody je mnoho. Jedním z nich je řešení elektrických obvodů pomocí diferenciálních rovnic.

V 2 kapitole se zaměříme na simulační systém TKSL/C(TKSL/386). Seznámíme se s jeho metodou řešení diferenciálních rovnic pomocí Taylorovy řady a řekneme si, jaké typy úloh můžeme s jeho pomocí řešit.

Ukážeme si způsob jak řesit elektrické obvody u kterých je sestavení diferenciálních rovnic problematické. Přidáním parazitních indukčností do obvodu má za důsledek snadnější popis obvodu a jejich vliv na obvod samotný není nijak znatelný. Takovou metodu můžeme nazvat Metodou parazitní indukčnosti při řešení elektrických obvodů.

Předchůdce TKSL/C má implementované grafické rozhraní pro zobrazení výsledných grafů diferenciálních rovnic popisujících simulovaný eletrický obvod. Další fází vývoje TKSL/C je implementace grafického systému. Cílem je vytvořit grafický systém, který bude kompatibilní s nově vznikající verzí TKSL. Jelikož nová implementace TKSL není plně funkční, bude nám prozatím jako zdroj dat vyhovovat TKSL/C. Výstupním jazykem systému TKSL/C je množina řetezců, kterou můžeme chápat jako vstupní data pro náš grafický systém (viz. 2.5). V kapitole 5 následuje popis návrhu samotného grafického systému.

V dalších kapitolách jsou popsány implementační záležitosti a problémy, se kterými jsem se během práce setkal. Nechybí ani ověření funkčnosti se systémem TKSL/C.

Závěrečné shrnutí je v kapitole 8.1. Uživatelská příručka k hotovému grafickému systému a obsah přiloženého CD jsou v příloze A a B.

Kapitola 2

TKSL

2.1 TKSL/C

K obecnému řešení spojitých simulačních úloh ve výpočetní technice se přistupuje zatím jen numericky, nikoliv analyticky. To znamená, že je potřeba vyčíslit stav modelu během simulace nějakou vhodnou numerickou metodou. Touto metodou je velmi často numerická integrační metoda, protože abstraktní model simulovaného systému bývá popsán soustavou diferenciálních rovnic. Kvalita celého simulačního systému závisí na kvalitě algoritmu numerické integrační metody. Ta totiž tvoří jádro výpočtu. Jejími kritickými vlastnostmi jsou stabilita, přesnost a rychlost. TKSL/C je postaven na numerické integrační metodě Taylorovy řady. A právě proto nabízí TKSL/C velmi stabilní, přesné a rychlé řešení diferenciálních rovnic a tím i spojitých simulačních úloh.

2.2 TKSL/386

Simulační jazyk TKSL/386, který je předchůdcem systému TKSL/C, byl vytvořený pro testování algoritmů využívajících k řešení diferenciálních rovnic a ostatních problémů Taylorovy řady. Tento systém byl vytvořený v uživatelsky přátelském prostředí TurboVision. Dovoluje uživateli nastavit přesnost výpočtu a řád metody. Zabezpečuje přesnou detekci nespojitostí a výpočet probíhá s proměnným integračním krokem. Výhodou systému je možnost použití schématu nakresleného v systému ORCAD. Další možnost je přímé využití integrovaného editoru zdrojového kódu pro překladač. Systém TKSL/386 disponuje překladačem jazyka vhodného pro jednoduchý popis diferenciálních rovnic. Po úspěšném přeložení kódu popisujícího analyzovaný systém je možné spustit simulaci. Průběhy řešení jsou přehledně zobrazené v grafu a pomocí kurzoru se dá zobrazit hodnota řešení ve vyznačených bodech.

2.3 Metoda Taylorovy řady

Tato metoda se osvědčila jako extrémně rychlá a přesná metoda numerické integrace[4]. Je dostatečně obecná, může řešit obyčejné diferenciální rovnice, soustavu diferenciálních rovnic, systémy s nespojitostmi, ale také tuhé systémy, kde tato metoda předčí své současníky svou stabilitou a přesností.

Umožňuje také řešení parciálních diferenciálních rovnic. Sestrojíme-li totiž nad oblastí, ve které chceme parciální dif. rovnice řešit, pravoúhlou síť a v každém uzlu této sítě na-

jdeme dif. rovnici, která charakterizuje podstatné vlastnosti tohoto uzlu, dostaneme soustavu mnoha diferenciálních rovnic, která je již metodou Taylorovy řady řešitelná. Reálným omezením zde může být výpočetní náročnost takového postupu.

Jak je z názvu patrné, tato metoda vychází z Taylorovy řady, která je poměrně přesná při výpočtu hodnoty dalšího kroku numerického řešení diferenciální rovnice. Taylorova řada používá hodnotu funkce vypočtenou v předchozím kroku pro výpočet nové hodnoty funkce, jak vyplývá ze vztahu 2.1.

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) + \frac{h^2}{2!} \cdot f'(t_n, y_n) + \cdots + \frac{h^p}{p!} \cdot f^{[p-1]}(t_n, y_n) \tag{2.1}$$

Proč se tedy nepoužívala Taylorova řada pro numerické řešení diferenciálních rovnic dříve? Problémem je výpočet dostatečně dlouhého rozvoje Taylorovy řady, tedy generování vyšších řádů derivace $f^{[1]}$, $f^{[2]}$, $\ldots$ $f^{[p-1]}$ tak, abychom obdrželi požadovanou přesnost výsledku. To je důvod, proč se obecně používá metoda Runge-Kutta. Pokud se nám však podaří získat derivace vyššího řádu, dostaneme velmi přesné výsledky omezené třeba jen přesností použitého modelu čísla v počítačové aritmetice.

Metoda Taylorovy řady má ještě jednu podstatnou výhodu. Díky členění Taylorova rozvoje se dá výhodným způsobem paralelizovat, a tak dosáhnout podstatně vyššího výpočetního výkonu.

2.4 Použití TKSL/386

Mějme jednoduchý obvod ve kterém jsou sériově zapojeny zdroj, odpor, cívka a kondenzátor[1]. U tohoto obvodu víme, že součet všech jmenovitých napětí na spotřebičích se rovná napětí na zdroji. Celý obvod můžeme popsat rovnicí

$$uL + uR + uC = u \tag{2.2}$$

nebo taky

$$L\frac{di}{dt} + Ri + \frac{1}{C}\int i dt = u \tag{2.3}$$

Po dosazení

$$y' = i \tag{2.4}$$

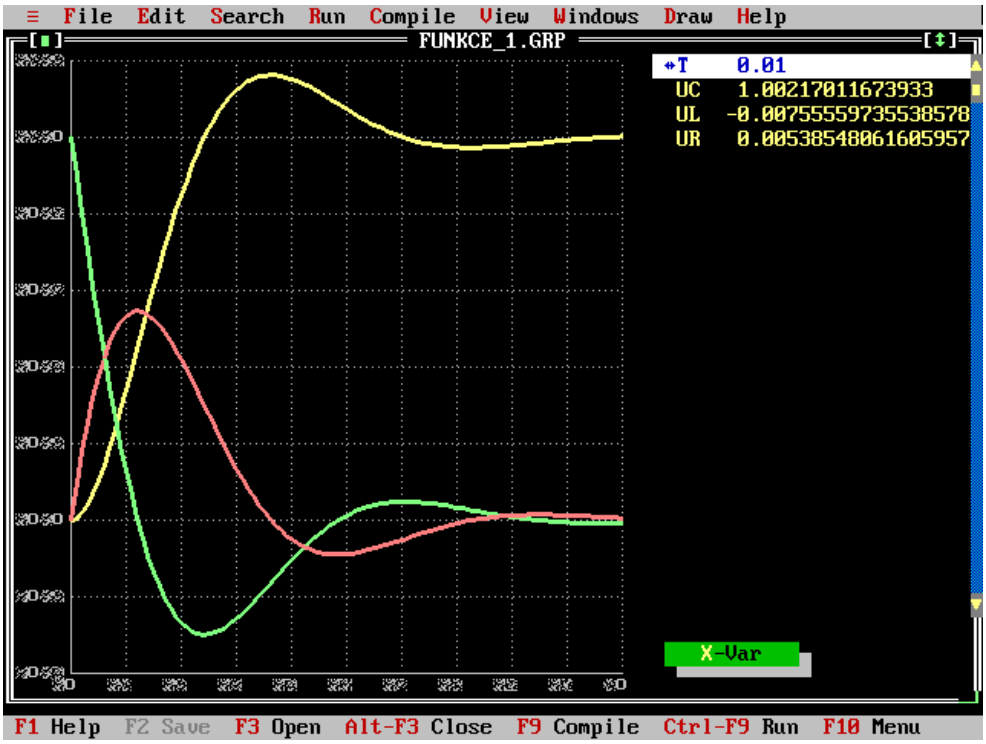
dostaneme

$$i' = \frac{1}{L}(u - Ri - \frac{1}{C}y) \tag{2.5}$$

Budeme počítat s hodnotami pro U = 1 V, R = 1000 Ω, L = 1 H, C = 10^{-6} F.
Pro TKSL/386 použijeme následující kód.

```
var i,y,u,UR,UL,UC;
const  R=1000,L=1,C=1E-6,tmax=0.01,dt=0.0001;
system
u=1;
i'=1/L*(u-R*i-1/C*y)    &0;
y'=i                    &0;
UR=R*i;
UC=1/C*y;
UL=u-R*i-1/C*y;
sysend.
```

Výsledný graf časových funkcí uR, uL a uC je zobrazen na obrázku 2.1.



Obrázek 2.1: Výstup simulace el. obvodu v TKSL/386

2.5 Výstupní jazyk TKSL/C

Výstupním jazykem TKSL/C rozumíme množinu všech řetězců generovanou programem a je výsledkem numerického výpočtu. Výstupní jazyk je tvořen sloupci. V každém sloupci se nachází samostatná proměnná (v prvním řádku je její název).
Příklad výstupních dat:

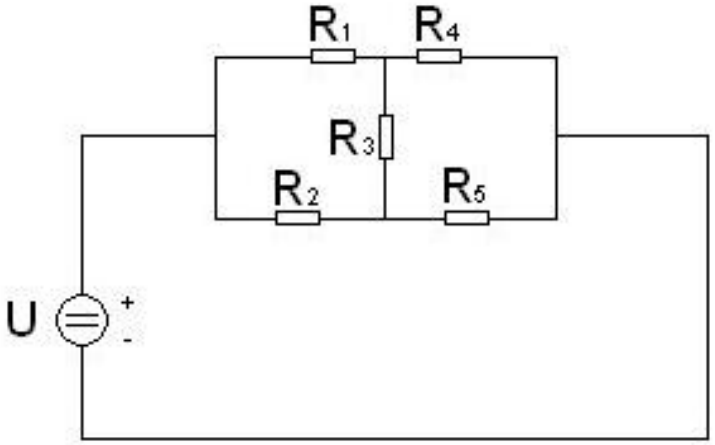
```
t x y #Order
0.00 1.0000000000 1.0000000000 0
0.01 0.9900499988 1.0100499987 4
0.02 0.9801999802 1.0201999798 4
0.03 0.9704499004 1.0304498971 4
0.04 0.9607996868 1.0407996732 4
0.05 0.9512492395 1.0512491979 4
0.06 0.9417984316 1.0617983280 4
0.07 0.9324471102 1.0724468863 4
0.08 0.9231950971 1.0831946607 4
```

Kapitola 3

Parazitní indukčnosti při řešení elektrických obvodů

V této kapitole si na jednoduchém elektrickém obvodu ukážeme řešení pomocí diferenciálních rovnic a parazitních indukčností. Ukážeme si jak kapacity do obvodu zapojit a jak odvodit výsledné rovnice pro simulační nástroj TKSL/386. V další části si ověříme funkčnost této metody a správnost výsledků. Pro kontrolu vyřešíme náš obvod pomocí klasické metody, kterou známe ze střední školy.

Zvolíme jednoduchý elektrický obvod, který bude obsahovat jen rezistory. Pokud bychom zapojili do obvodu nelineární prvky, tak by se v případě klasické metody muselo počítat pomocí komplexních čísel.



Obrázek 3.1: Schéma jednoduchého obvodu

3.1 Výpočet pomocí parazitních indukčností

3.1.1 Zjištění proudových smyček

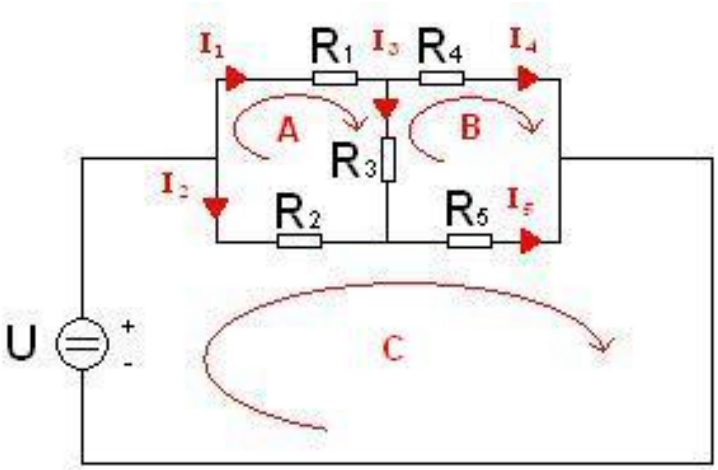
Abychom mohli obvod vyřešit je potřeba do něj umístit parazitní indukčnosti (cívky) a to tak, aby každá smyčka obsahovala právě jednu. Kolik je v obvodu smyček a které to jsou

zjistíme pomocí Kirchhoffových zákonů, a to konkrétně podle druhého zákona (o napětích, o smyčkách). Druhý Kirchhoffův zákon říká, že:

Zákon 3.1.1 *Součet úbytků napětí na spotřebičích se v uzavřené části obvodu (smyčce) rovná součtu elektromotorických napětí zdrojů v této části obvodu.*

(Pokud by zákon pro nějakou smyčku neplatil, mohlo by být sestrojeno perpetuum mobile, ve kterém by proud touto smyčkou procházel neustále dokola při permanentním odběru energie.)
Postup při řešení elektrického obvodu pomocí metody smyčkových proudů je následující:

- Na schématu se najdou elementární smyčky, tzn. smyčky, které neobsahují menší vnořené smyčky.
- Každé takové smyčce se přidělí proud, který jí obíhá.
- Pro každou smyčku se zapíše rovnice podle 2. Kirchhoffova zákona, ve které se jako neznámá použije proud protékající smyčkou.
- Tato soustava rovnic se poté vyřeší.



Obrázek 3.2: Schéma obvodu s proudy

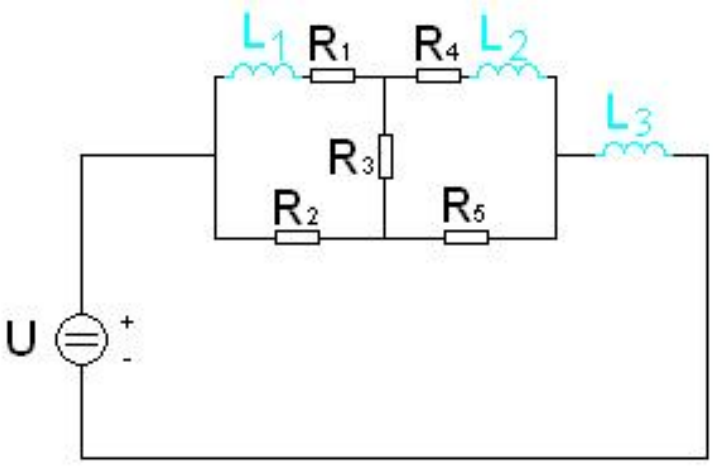
Na obrázku 3.2 vidíme proudy, které jsou předmětem našeho řešení. Jsou to proudy I_1, I_2, I_3, I_4 a I_5. Dále jsou zde vykresleny všechny elementární smyčky našeho obvodu. Podle postupu každé takové smyčce přidělíme proud, který ji obíhá. Smyčkou A probíhá proud I_A, smyčkou B proud I_B a smyčkou C proud I_C. Na volbě směru oběhu nezáleží (volím ve směru hodinových ručiček), ale je nutné tento směr během výpočtu dodržet.
V této fázi výpočtu už známe jednotlivé vztahy pro výpočet všech proudů v obvodu.

$$\begin{aligned} I_1 &= I_A \\ I_2 &= I_A - I_C \\ I_3 &= I_A - I_B \\ I_4 &= I_B \\ I_5 &= I_B - I_C \end{aligned} \tag{3.1}$$

Proud v konkrétní větvi vypočítáme jako součet všech sousedních proudů smyček. Pokud je směr oběhu opačný, je proud záporný. V tomto případě se nám příklad přehodnotil na řešení pouze proudů obíhajících ve smyčkách A, B a C. Pro výpočet těchto proudů použijeme parazitní indukčnosti. Parazitní indukčnosti se vhodně připojí do našeho obvodu (Obrázek 3.3). Jakmile se obvod zapne, proud na cívkách se začne zvětšovat a po určitém čase se ustálí na maximální hodnotě. Tomuto stavu říkáme ustálený stav. Výsledné proudy jsou naše proudy I_A, I_B a I_C.

Zbývá vytvořit diferenciální rovnice pro výpočet smyčkových proudů. Diferenciální rovnice pro výpočet proudu na cívce vyplývá ze vztahu 3.2.

$$i' = \frac{1}{L}u \tag{3.2}$$



Obrázek 3.3: Schéma obvodu s parazitními indukčnostmi

3.1.2 Odvození rovnic smyček

Podle Kirchhoffových zákonů (viz. 3.1.1) si sestavíme rovnice pro všechny tři smyčky.

$$\begin{aligned} L_1 i'_A + R_1 i_A + R_3(i_A - i_B) + R_2(i_A - i_C) &= 0 \\ R_4 i_B + L_2 i'_B + R_5(i_B - i_C) + R_3(i_B - i_A) &= 0 \\ R_2(i_C - i_A) + R_5(i_C - i_B) + L_3 i'_C - u &= 0 \end{aligned} \tag{3.3}$$

Po dosazení za i'_A, i'_B a i'_C ze vztahu 3.2 dostaneme rovnice pro jednotlivé proudy.

$$\begin{aligned} i'_A &= -\frac{1}{L1}(R1 i_A + R3(i_A - i_B) + R2(i_A - i_C)) \\ i'_B &= -\frac{1}{L2}(R4 i_B + R5(i_B - i_C) + R3(i_B - i_A)) \\ i'_C &= -\frac{1}{L3}(R2(i_C - i_A) + R5(i_C - i_B) - u) \end{aligned} \tag{3.4}$$

3.1.3 Výpočet v TKSL/386

V této části si náš obvod odsimulujeme a uvidíme jak je výpočet přesný. Rovnice, které jsme si odvodili, je potřeba převést do formátu programu TKSL/386. Obvod budeme počítat s hodnotami pro $R_1 = 200\Omega$, $R_2 = 300\Omega$, $R_3 = 500\Omega$, $R_4 = 100\Omega$, $R_5 = 50\Omega$, U = 32 V, $L_1 = 10^{-12}$ H, $L_2 = 10^{-12}$ H a $L_3 = 10^{-12}$ H.

Zdrojový kód pro TKSL/386:

```
var iA, iB, iC, i1, i2, i3, i4, i5;
const dt=1e-15,tmax=1e-13,eps=1e-20,
u = 32, R1=200, R2=300,R3=500, R4=100,R5=50, L1=1e-12, L2=1e-12, L3=1e-12;

system
iA'= 1/L1*(- R1*iA - R3*iA + R3*iB - R2*iA + R2*iC)    &0;
iB'= 1/L2*(- R4*iB - R5*iB + R5*iC - R3*iB + R3*iA)    &0;
iC'= 1/L3*(- R2*iC + R2*iA - R5*iC + R5*iB + u)        &0;

i1 = iA;
i2 = iC - iA;
i3 = iA - iB;
i4 = iB;
i5 = iC - iB;
sysend.
```

Na Obrázku 3.4 vidíme ustálený stav proudů ve smyčkách. Můžeme vidět, jak se proud na cívce postupně zvedal až se ustálil na své maximální hodnotě.

Obrázek 3.4: Proudy ve smyčkách

Obrázek 3.5: Vypočítané proudy v obvodu

Při výpočtu proudů v obvodu se může stát, že nám nějaký proud vyjde záporně (Obrázek 3.5). To je způsobeno volbou směru smyčkových proudů. V takovém případě stačí hodnotu převést na kladnou.

3.1.4 Stiff systémy

Použití parazitních indukčností má i své problémy. Pokud chceme připojit k obvodu parazitní indukčnost, tak by měla být dostatečně malá. Měla by mít oproti prvkům v obvodu o několik řádů menší hodnotu indukčnosti. Důvodem je, aby indukčnost změnila chování obvodu jen tak minimálně, aby se to dalo zanedbat. A to je problém při řešení diferenciální rovnice, ve které jsou nelineární prvky, jejichž parametry se liší až v několika řádech. Takové diferenciální systémy potřebují na vyřešení prvků s nízkou hodnotou velmi malý krok při řešení pomocí Taylorovy řady. Na druhé straně jsou tam prvky s vysokou hodnotou, které budou mít dlouhý simulační čas. Takže v konečném výsledku bude simulace trvat velmi dlouhou dobu. Takové systémy nazýváme Stiff systémy[2].

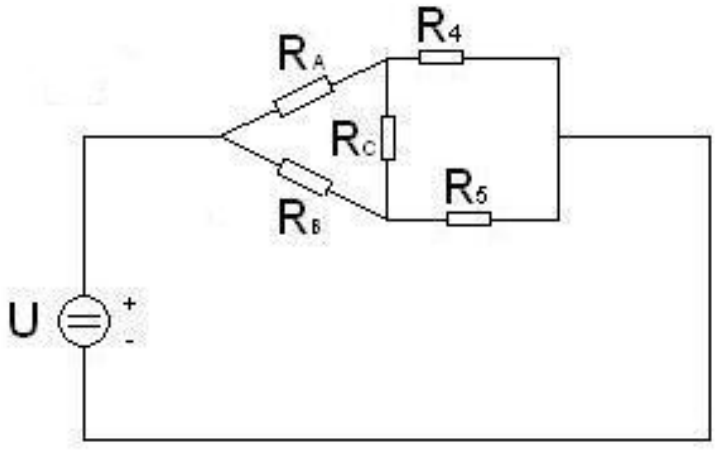
3.2 Výpočet pomocí klasické metody

V podkapitole 3.1 jsme si ukázali, jak elegantně můžeme vyřešit elektrický obvod, když do něj zapojíme parazitní indukčnosti. V této části si zkusíme náš obvod z obrázku 3.1 vyřešit pomocí klasické metody. Náš obvod nebudeme nijak upravovat. Všechny hodnoty zůstanou stejné jako u předchozí metody.

3.2.1 Řešení obvodu

Pokud se podrobně podíváme na obvod, zjistíme, že obsahuje několik paralelně a sériově zapojených rezistorů. Bohužel pro jednoduché použití Ohmova zákona nám brání rezistor R_3.

Zkusme si náš obvod trochu překreslit.



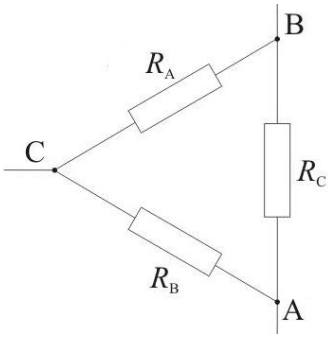
Obrázek 3.6: Překreslený obvod s trojúhelníkem

Na obrázku 3.6 vidíme jak obvod vypadá po překreslení. Vznikl nám zde nový útvar, elektrický trojpól. Takový elektrický útvar se jmenuje trojúhelník.

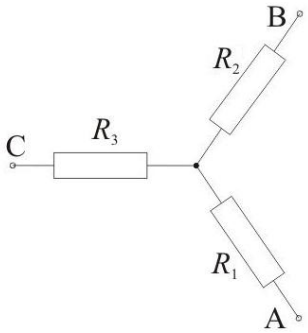
Trojúhelník a jeho transformace na hvězdu

Elektrické trojpóly mají tu vlastnost, že je můžeme mezi sebou libovolně transformovat. Při převodu je důležité dodržet jedno pravidlo:

Pravidlo 3.2.1 *Aby se obě zapojení chovala stejně, musí být mezi každými dvěma body v obou zapojeních stejný celkový odpor.*

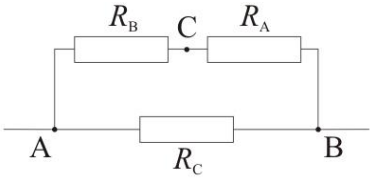


Obrázek 3.7: Trojúhelník



Obrázek 3.8: Hvězda

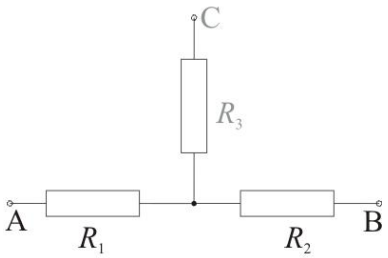
Pokud v našem obvodu vzniklý trojúhelník transformujeme na hvězdu, vznikne obvod, který dokážeme jednoduše vyřešit. Mezi každými dvěma body musí být při obou zapojeních stejný celkový odpor. Pro odpor mezi body A a B v zapojení do trojúhelníku si můžeme obvod překreslit takto:



Obrázek 3.9: Překreslený trojúhelník

$$\begin{aligned} \frac{1}{R_{AB}} &= \frac{1}{R_C} + \frac{1}{R_B + R_A} \\ \frac{1}{R_{AB}} &= \frac{(R_B + R_A) + R_C}{R_C(R_B + R_A)} \\ R_{AB} &= \frac{R_C(R_A + R_B)}{R_A + R_B + R_C} \end{aligned} \tag{3.5}$$

Obvod pro odpor mezi body A a B pro hvězdu můžeme nakreslit takto:



Obrázek 3.10: Překreslená hvězda

Z předešlého obrázku tedy vyplývá

$$R_{AB} = R_1 + R_2 \tag{3.6}$$

Po dosazení za R_{AB} ze vztahu 3.5 získáme rovnici

$$R_1 + R_2 = \frac{R_C(R_A + R_B)}{R_A + R_B + R_C} \tag{3.7}$$

Jelikož jsou obvody symetrické, stačí pro odvození zbylých dvou rovnic zaměnit indexy.

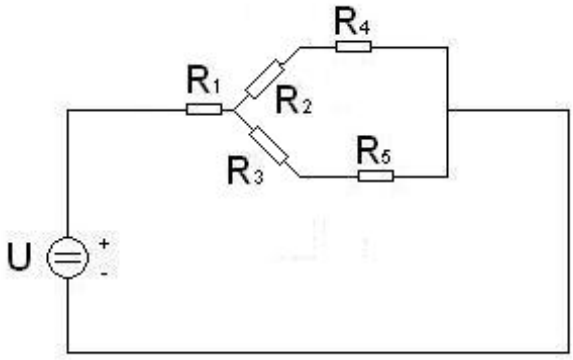
$$\begin{aligned} R_1 + R_3 &= \frac{R_B(R_A + R_C)}{R_A + R_B + R_C} \\ R_2 + R_3 &= \frac{R_A(R_B + R_C)}{R_A + R_B + R_C} \end{aligned} \tag{3.8}$$

Výsledkem vzniklé soustavy tří rovnic o třech neznámých jsou následující vztahy, které vrací velikosti odporů po transformaci trojúhelníku na hvězdu.

$$
\begin{aligned}
R_1 &= \frac{(R_B R_C)}{R_A + R_B + R_C} \\
R_2 &= \frac{(R_A R_C)}{R_A + R_B + R_C} \\
R_3 &= \frac{(R_A R_B)}{R_A + R_B + R_C}
\end{aligned} \tag{3.9}
$$

Výpočet

Transformací trojúhelníku na hvězdu vznikl obvod, který vidíme na obrázku 3.11. Teď jej můžeme lehce spočítat pomocí Ohmova zákona.



Obrázek 3.11: Transformovaný obvod na hvězdu

Odpor R_{2345} získáme jako paralelní spojení odporů R_{24} a R_{35}. Dosadíme do vztahu na výpočet dvou paralelně zapojených odporů

$$
\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \tag{3.10}
$$

a získáme rovnici pro odpor R_{2345}

$$
R_{2345} = \frac{R_{24} R_{35}}{R_{24} + R_{35}} \tag{3.11}
$$

Výsledný odpor získáme sériovým zapojením odporů R_1 a R_{2345}. Nakonec dosadíme do Ohmova zákona a tím získáme proud, který obvodem protéká.

$$
I = \frac{U}{R} \tag{3.12}
$$

Kapitola 4

Grafické systémy

Systémy pro modelování a simulace vznikaly už v dobách prvních počítačů, kdy jeden počítač zabíral celou budovu a měl v sobě tisíce elektronek. Simulace na počítači šetří čas i peníze. Kolikrát se simulují modely reálných objektů pro člověka nebezpečných jako třeba náraz auta nebo exploze. Někdy je nemožné provést simulaci v reálu, protože samotný předmět simulace je mimo lidské možnosti (simulace oběhu planet) nebo čas samotného pozorování je řádově roky. Tenkrát ještě neexistovalo grafické rozhraní, takže výstupem nebylo nic víc než čísla. Jak se postupně s časem technika zdokonalovala a rozšiřovala, rostly nároky na zpracování výsledků simulace. Předpokladem byla názornost, přehlednost a jednoduchost. V současnosti se zpracování výsledků snaží napodobit to, co je nám blízké. Moderní 3-D počítačová grafika je jistě příjemnější.

V této kapitole se zamyslíme co by měl moderní grafický systém obsahovat a jak být co nejvíce využitelný.

4.1 Vlastnosti grafického systémů

4.1.1 Ovládání

Cílem asi všech systémů je jednoduché a pohodlné ovládání. Reakce uživatelů na ovládání je předmětem mnoha studií (MS Office 2008). V moderních systémech je ovládání koncipované podle druhu akce, kterou chceme vykonávat a podle toho si můžeme systém přizpůsobit.

Zpracování detailu

Tato funkce, anglicky „zoom“, je často využívaná v grafických systémech pro zobrazení části grafu nebo kreslící oblasti. K výběru se převážně používá výřez označený myší. Pro vykreslení detailu se pak použije celá původní kreslící oblast. Systémy poskytují uživateli také možnost vrátit se zpět k původnímu detailu nebo zobrazit celý graf na celou kreslící oblast.

Volba barvy

Pro rozlišení jednotlivých průběhů grafických křivek je vhodné použít různé druhy barev. Uživatel by měl mít možnost nastavit vlastní požadovanou barvu. Obvykle jsou ale barvy přiřazovány grafickým systémem a v tomto případě je důležitá volba barev. Systém by měl volit takové barvy, které jsou kontrastní vůči pozadí a zároveň vůči sobě. Převážně se jedná o barvy základní.

Jednobarevné zobrazení

Kvalitní systém obsahuje možnost zobrazit výsledné grafické křivky nejenom v různých barvách, ale i tvarech. Tisk křivek na černobíle tiskárně, která podporuje jen odstíny šedi, je nepraktický a nepřehledný. Proto by systém měl být obohacen o možnost změny tloušťky čáry nebo tvaru. Např. čárkovaná, přerušovaná nebo tečkovaná. Možné je i použití grafické značky (kolečko, čtverec) po určitém úseku.

4.1.2 Rychlost

Nejenom hardwarové vybavení udává rychlost na něm běžícím systému. Je zde snaha o zvýšení rychlosti použitím vhodných algoritmů, které se stejným výsledkem zaberou méně paměti nebo systémového času.

Průběžné vykreslování

Průběžným rozumíme takové vykreslování, při němž je vykreslován výsledný graf již během načítání hodnot funkce. Díky této vlastnosti může uživatel včas pozastavit výpočet, neodpovídají-li získané průběžné výsledky jeho představám.

4.1.3 Zpracování většího množství dat

Snaha o paralelizaci a rozdělení výpočtu mezi větší počet počítačů je dobrým způsobem, jak zpracovat více operací v jednom cyklu.

4.1.4 Prostorové zobrazení výsledků

Možnost zobrazení výsledku výpočtu pomocí 3-D počítačové grafiky je vhodné použít tam, kde je nám prostorové vnímání bližší. Je hojně využívané například v aplikacích pro modelování ve strojírenství nebo architektuře.

4.1.5 Názorné zobrazení vypočtených dat

Přehledné a přitom maximálně názorné zobrazení s možností konverze do jiných standardizovaných formátů.

4.1.6 Podpora sdílení dat

Nedílnou součástí grafických systémů je i podpora sdílení a možnost prezentace dat mezi jinými uživateli v rámci celosvětové sítě Internet.

Kapitola 5

Návrh grafického systému

V této kapitole se zaměříme na návrh grafického výstupu pro simulační nástroj TKSL. V minulé kapitole jsme se dozvěděli, co všechno by měl grafický systém obsahovat a teď se na některé tyto části zaměříme a probereme je podrobněji.

5.1 Vývojové prostředí

Pro návrh aplikací existuje mnoho vývojových prostředí. Tato vývojová prostředí jsou závislá především na programovacím jazyce a částečně na operačním systému. Pro tvorbu této práce byl zvolen jazyk C# z balíku .NET a jako vývojová platforma Windows.

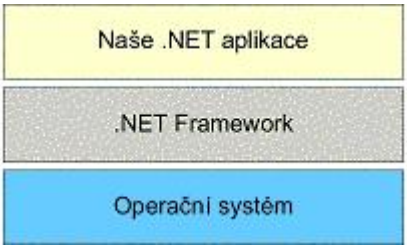
5.1.1 .NET

Součástí skupiny .NET[5], která je dostupná pro Windows, dále jako webové rozhraní a jako platforma pro Pocket PC jsou programovací jazyky VB, C++, J#, C#. Těmito jazyky napsané aplikace můžeme bez problémů převést do ASP.NET (Web) nebo do aplikace pro Pocket PC (.NET Compact Framework).Pro tvorbu aplikací spojující tyto myšlenky vydal Microsoft Visual Studio .NET, které bylo oproti předešlé verzi rozšířené o jednoduchý návrh webových XML služeb a o .NET Framework, zajišťující prostředí pro běh aplikací a fungující jako spouštěcí rozhraní a jako zdroj potřebných knihoven.

.NET Framework je nadstavba nad operačním systémem přicházející s novou vývojářskou platformou založenou na otevřených Internetových protokolech, poskytujících aplikacím, službám a zařízením schopnost vzájemně spolupracovat. Je to balík knihoven, jehož jednotné prostředí nabízí své objektové třídy vyšším programovacím jazykům, aby jej využily k práci při řešení konkrétních úloh nezávisle na použitém programovacím jazyce. Vztah .NET Frameworku k okolí bude nejlépe patrný z následujícího schématu:

.NET Frameworku je již jedno, jaký programovací jazyk ovládáme, protože pro všechny .NET jazyky nabízí stejné rozhraní. .NET Framework je pro majitele operačního systému Windows k dispozici zdarma jako samostatná komponenta, která se do systému doinstaluje. Stejné objektové třídy se stejnými metodami a vlastnostmi (až na drobné výjimky) jsou nyní dostupné jak programátorům ve Visual Basicu tak třeba v novém C#.

Jelikož .NET vyvíjí Microsoft jako komerční produkt, jsou tyto technologie dobře dostupné. K dispozici je i verze .NET Compact Framework (.NET CF) pro Pocket PC s operačním systémem Windows Mobile, která je s klasickou verzí kompatibilní, a tak není nutné kompilovat různé aplikace pro PC a Pocket PC - na obou systémech bude aplikace fungovat stejně.



Obrázek 5.1: Vztah .NET Frameworku k okolí

GNU obdoba .NET se nazývá DotGNU a umožňuje spustit všechny .NET aplikace na unixových platformách.

5.2 Návrh

Jelikož programovací jazyk C# je objektový, bude i tento návrh a implementace objektově orientovaná. Při návrhu jsem čerpal z prostředí Matlab a TKSLGraph[4], kde je vykreslování výsledných grafů dobře propracováno.

Hlavní požadavky pro náš systém:

- systém umí vykreslit libovolné množství grafických křivek,
- každá z těchto křivek bude mít možnost zobrazení nebo skrytí,
- u křivek se budou dát změnit základní parametry jako barva, popisek, viditelnost a zdroj dat pro horizontální a vertikální osy,
- automatické vykreslení os v závislosti na oboru hodnot,
- možnost vykreslení os souřadného systému a zobrazení mřížky,
- jednoduchý export grafu jako bitmapa,

5.2.1 Spolupráce s TKSL/C

Jelikož pro tento systém zatím není žádný generovaný zdroj vstupních dat (budoucí verze TKSL), používá se jako vzorová data výstup ze starší verze simulačního nástroje TKSL/C. Výstupní jazyk TKSL/C byl popsán v podkapitole 2.5. Při načítání dat z tohoto jazyka se nejprve zpracuje první řádek (hlavička) a zjistí se kolik obsahuje názvů proměnných. Předpokládá se, že na každém dalším řádku bude stejný počet řetězců reprezentujících číslo jako proměnných na prvním řádku.

5.2.2 Uschování načtených dat

Pro načtení dat je v systému potřeba vytvořit nějakou vhodnou strukturu, která v sobě uchová veškeré potřebné informace. Předem nevíme kolik hodnot resp. zdrojů dat pro křivky budeme ukládat a ani nevíme kolik tyto zdroje obsahují vzorků. Proto tato struktura musí být navržena tak, aby se její obsah dal dynamicky měnit. Zjednodušeně naši strukturu můžeme popsat jako delegaci třídy zdrojů dat pro křivky. Každý zdroj obsahuje jméno zdroje a všechny vzorky. Při ukládání vzorků do zdroje se automaticky vyhodnocuje nejmenší a největší vzorek.

5.2.3 Křivka

Křivka v sobě ponese informace o zdroji dat, a to pro vertikální a horizontální obor hodnot. Kdykoliv bude možné tyto hodnoty změnit. Dále je potřeba vědět jestli se má přímka zobrazovat a jakou barvou se má vykreslit. V případě skrytí bude potřeba ověřit, jestli se změní rozsah os a jestli je nutné je překreslit.

5.2.4 Osy

Výpočet bodů na horizontální a vertikální ose by měl být automatický a měl by se při jakékoliv změně oboru hodnot překreslit. Změnou chápeme přidání nebo upravení křivky do systému nebo zvětšení či zmenšení zobrazovací plochy. Pokud se zobrazovací plocha zvětší resp. zmenší, je potřeba počet bodů na ose také zvětšit resp. zmenšit. Pro přehlednější zobrazení je dobré zobrazovat osy souřadných os a taky mít možnost jejich zobrazení vypnout. Užitečné je i možnost zobrazení a skrytí mřížky.

Kapitola 6

Implementace

6.1 Křivka

Implementace křivky je popsaná třídou `GraphLine` a ta obsahuje:

- jméno křivky
- viditelnost
- barvu
- referenci na horizontální zdroj dat
- referenci na vertikální zdroj dat

Při spuštění programu jsou tyto přímky automaticky vytvořeny ze zdrojových dat a to tak, že zdroj dat, který je v souboru (výstupní jazyk TKSL/C) v prvním „sloupci“, je použitý jako horizontální zdroj dat. Pro ostatní zdroje dat se vytvoří nové křivky, kde tyto zdroje reprezentují vertikální zdroj dat..
Pokud bude náš soubor vypadat jako vzorová ukázka výstupního jazyka TKSL/C (viz. 2.5), tak se do systému načtou 4 zdroje dat a vzniknou 3 křivky. První bude mít jméno x, horizontální zdroj bude zdroj t a vertikální zdroj bude zdroj x, atd.

Poznámka 6.1.1 *Pokud budeme systém používat jako modul do TKSL je vytváření křivek čistě v režiji TKSL a tuto metodu bude nutné doimplementovat.*

Křivku je možné upravit dodatečně (Obrázek 6.1). Umožňuje nám to třída `LineConfigForm`, která zastřešuje operace se všemi křivkami včetně vytvoření nové křivky. Tím, že můžeme u křivek dodatečně změnit zdroje dat, rozšiřujeme možnosti tohoto systému o zobrazení nejenom časových charakteristik ale i fázových nebo frekvenčních.

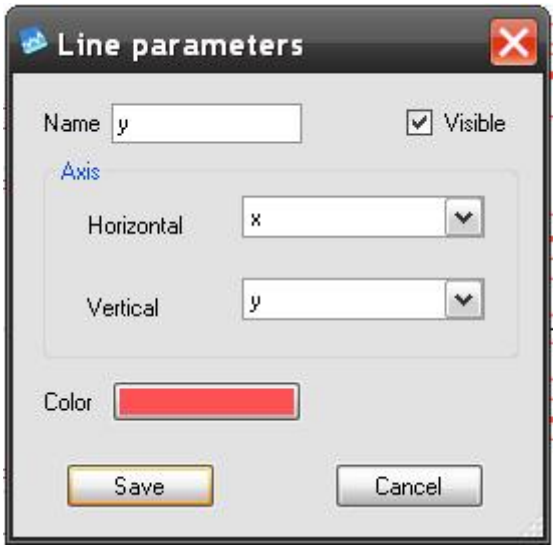
V případě změny zdroje dat u křivky nebo vytvoření nové křivky je třeba zjistit jestli není nutné přepočítat rozsah os.

Budeme mít křivku x a y. Křivka x a y bude mít horizontální zdroj dat z rozsahu daném množinou A.

$$A = \{x | x \in R; x \in < 0; 10 > [1]\} \tag{6.1}$$

Sjednocením těchto dvou množin je opět interval $< 0; 10 >$.

[1] Hodnoty intervalu lze získat za zdroje dat jako nejmenší a největší vzorek



Obrázek 6.1: Editace parametrů křivky

Pokud ale u křivky y tento zdroj zaměníme za rozsah daným množinou B

$$B = \{x | x \in R; x \in < -10; 0 >\} \tag{6.2}$$

bude sjednocení těchto množin

$$A \cup B \tag{6.3}$$

a výsledný interval se změní na $< -10; 10; >$, tím dojde ke změně rozsahu horizontální[2] osy a je ji potřeba překreslit.

6.2 Osy

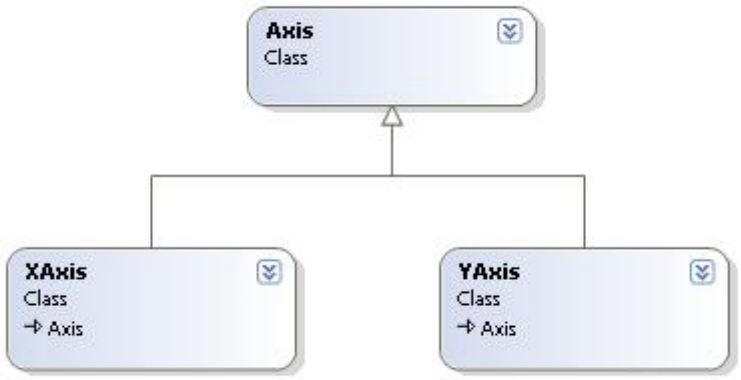
Jakým způsobem získáme množinu hodnot pro horizontální a vertikální osu jsme si ukázali v předchozí podkapitole. V této si ukážeme vlastní vykreslení os.

6.2.1 Vykreslení os

O zobrazení os se starají třídy `XAxis` a `YAxis`. Jak je patrné z obrázku 6.2 obě tyto třídy dědí od třídy `Axis`, která obsahuje několik virtuálních metod. Takové metody jsou pro nás užitečné, protože například metoda `DrawCoordWithValue()` se v obou třídách `XAxis` a `YAxis` jmenuje stejně (každá vykreslí na svoji osu na dané souřadnici čárku) ale jejich vnitřní implementace je rozdílná.

Postup pro vykreslení obou os je analogický. Budeme se tedy věnovat jen horizontální ose.
Důležité atributy této osy je šířka „okna“ do kterého kreslíme. Podle toho určujeme kolik bodů bude na ose umístěno.

[2] Analogicky platí pro vertikální osu



Obrázek 6.2: Dědičnost os

Body na ose

Na vykreslování bobů můžeme využít jednoduchý rekurzivní algoritmus.

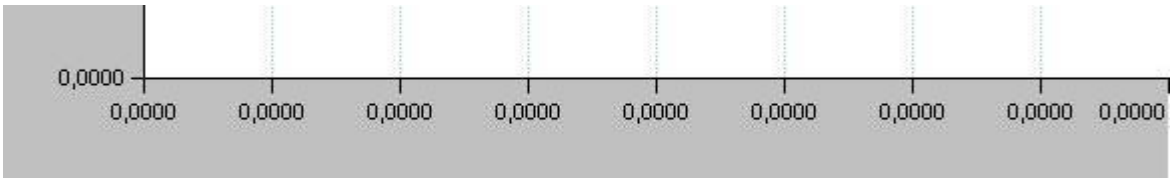
- Vykreslíme bod v nejmenší souřadnici okna a v souřadnici dané šířkou (největší) okna.
- Zavoláme funkci `ZjistiStred()` s parametry nejmenší souřadnice a největší souřadnice.
- Ve funkci zjistíme střed okna.
- Vykreslíme bod na souřadnici středu okna.
- Pokud je šířka polovičního okna větší něž dvojnásobek minimálního rozestupu mezi body[3], zavoláme na obě půlky oken rekurzivně `ZjistiStred()`.
- Pro první půlku s parametry nejmenší souřadnice a středu okna.
- Pro druhou půlku s parametry souřadnice středu a největší souřadnicí.

Tento algoritmus ještě obohatíme o zobrazení čísla k bodu, protože už víme jak zjistit množinu hodnot, která bude na naší ose. Tato množina je ohraničená dvěma čísly a to nejmenším a největším číslem z horizontálního zdroje dat. Přidáme tyto dvě čísla jako další dva parametry do naší funkce `ZjistiStred()` a po zavolání v ní zjistíme jejich aritmetický průměr a vykreslíme spolu a bodem. Parametry pro rekurzivní volání funkce přidáme k parametrům pro souřadnice.

Mřížka

Pokud chceme zobrazovat spolu s body i část mřížky(tyto úsečky jsou rovnoběžné s vertikální osou), stačí přidat k zobrazení bodu ještě vykreslení úsečky.

[3]Pro nás 200 pixelů



Obrázek 6.3: Vykreslení horizontální osy

Souřadná osa

Souřadná osa prochází na horizontální ose souřadnicí 0 a je kolmá k vertikální ose. Jelikož přesně nevíme, kde tato souřadnice leží, je nutné ji na ose přibližně vyhledat. Použijeme přitom náš upravený rekurzivní algoritmus.
Pro zjednodušení:
největší horizontální číslo = Hmax,
nejmenší horizontální číslo = Hmin,
největší souřadnice okna = Omax,
nejmenší souřadnice okna = Omin,

- Zavoláme funkci `NajdiCislo()`s parametry Omin, Omax, Hmin a Hmax.
- Ve funkci zjistíme střed okna a průměrnou hodnotu mezi čísli Hmax a Hmin.
- Pokud je průměrná hodnota čísel naše hledané číslo 0, tak vykreslíme osu a končíme.
- Pokud to není 0, tak porovnáme průměrnou hodnotu čísel s naším hledaným.
- Pokud je 0 větší, voláme rekurzivně `NajdiCislo()` s parametry střed okna, Omax, průměrná hodnota čísel a Hmax.
- Pokud je 0 menší, voláme rekurzivně `NajdiCislo()` s parametry Omin, střed okna, Hmin a průměrná hodnota čísel.

Zobrazení horizontální i vertikální osy je možné vypnout.

6.3 Zapojení modulu do aplikace

Zapojení modulu do aplikace je jednoduché. Celý modul je samostatný celek. Stačí vytvořit instanci třídy `Graph_Form`, což nám vytvoří nový formulář. Následně je třeba zavolat metodu `GetPointer()`, která vrátí odkaz na vytvořenou instanci třídy Spline. Do této instance je nutné uložit vypočítané hodnoty zdrojů dat. Poté stačí formulář spustit.

Formulář se vytvoří jako dialogové okno. To znamená že hlavní okno aplikace (hostitelský program) nebude aktivní, dokud uživatel dialog nezavře.

Kapitola 7

Použití systému

7.1 Modul pro TKSL

Tento grafický systém je primárně navržen jako modul pro zobrazování výstupních dat nově vznikající verze TKSL. Bohužel zatím není dostupná verze, která by poskytovala výstupní data pro generování grafů. Pro testovací účely je zde implementována možnost načíst data ze souboru. Vstupní jazyk dat systému je kompatibilní s výstupním jazykem řetězců, které generuje simulační nástroj TKSL/C. Ukázka výstupních dat je v podkapitole 2.5.

7.2 Ukázky

7.2.1 Sériový RLC Obvod

V podkapitole 2.4 jsme si ukázali použití simulačního nástroje TKSL/386 na vzorovém příkladu. Jednalo se o sériový RLC obvod, který byl popsán rovnicí

$$L\frac{di}{dt} + Ri + \frac{1}{C}\int idt = u \tag{7.1}$$

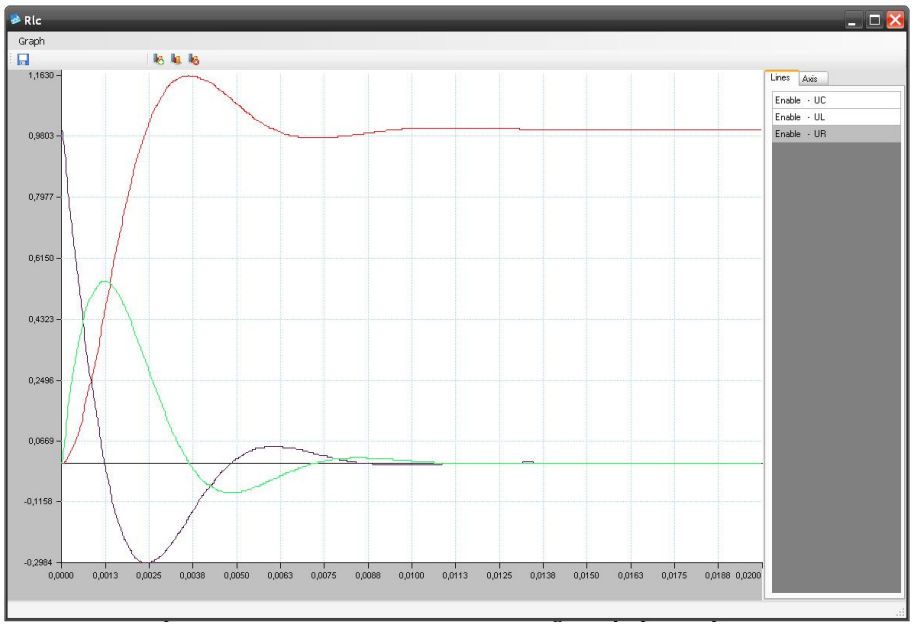
Výsledné časové průběhy napětí u_R, u_L a u_C jsou na obrázku č.7.1. Zdrojový soubor s daty, rlc.tksl, je možné najít na přiloženém CD ve složce bin/data.

7.2.2 Sinus a cosinus

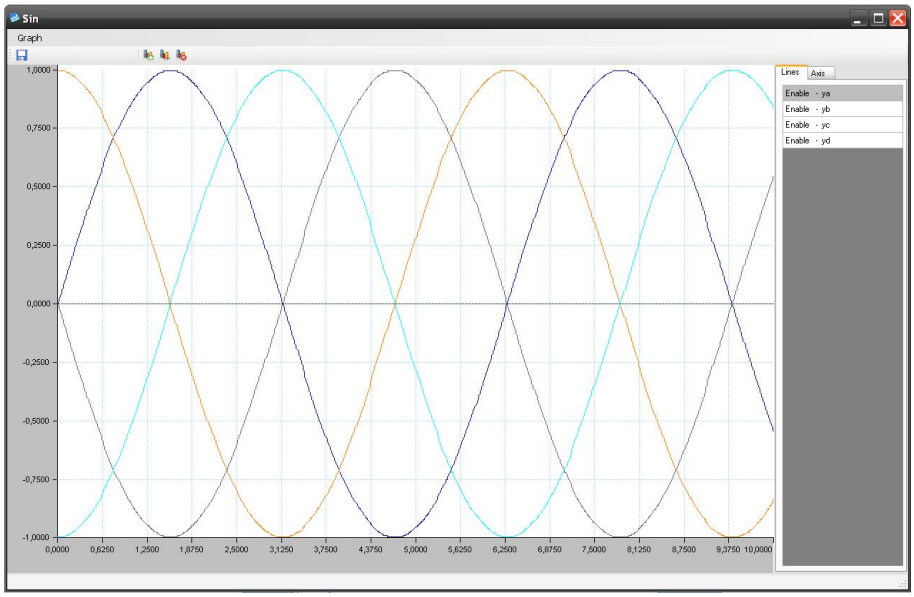
Ukázka zobrazení goniometrických funkcí sinus a cosinus(soubor sin.tksl).

$$\begin{aligned} xa &= sin(t) \\ xb &= sin(t - \frac{\pi}{2}) \\ ya &= cos(t) \\ yb &= cos(t - \frac{\pi}{2}) \end{aligned} \tag{7.2}$$

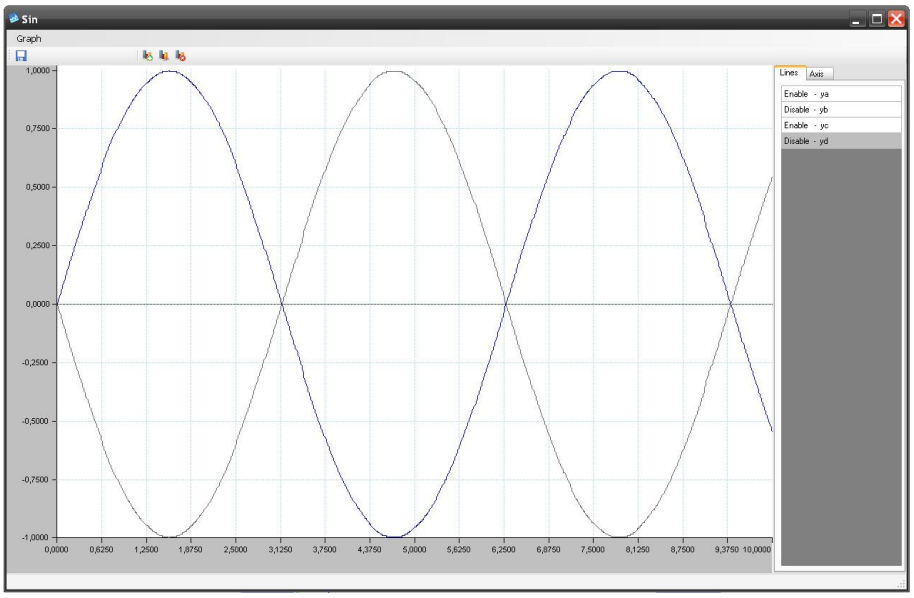
Na obrázku 7.2 jsou zobrazeny průběhy výše uvedených funkcí a na obrázku č.7.3 jen sinus a cosinus bez jejich posunutých funkcí.



Obrázek 7.1: Výsledné časové průběhy napětí v obvodu



Obrázek 7.2: Ukázka vykreslení goniometrických funkcí sinus a cosinus a jejich posunutých funkcí



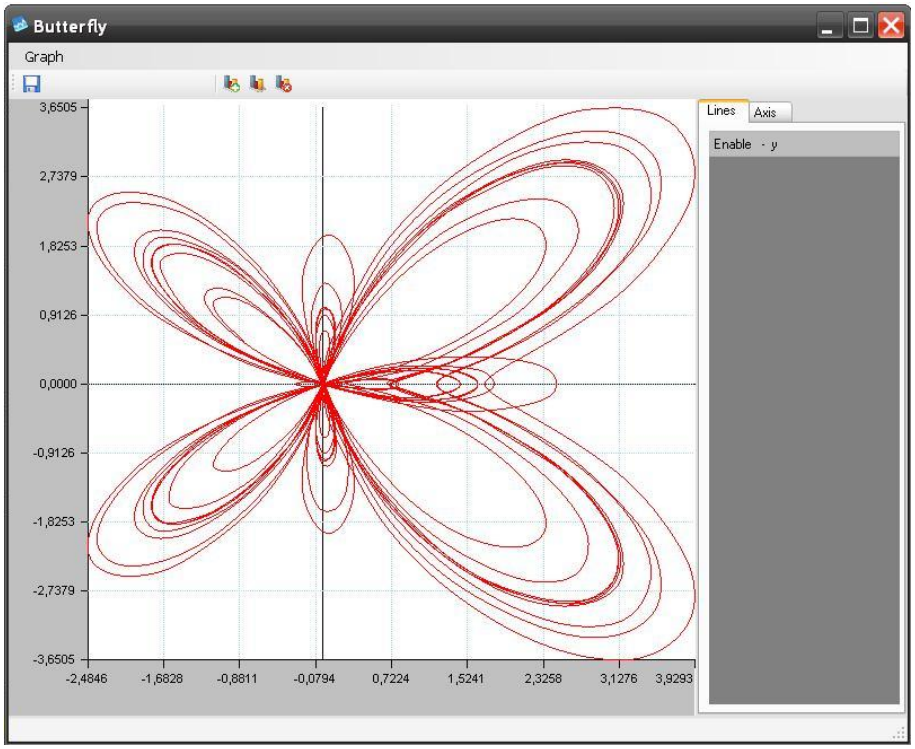
Obrázek 7.3: Ukázka vykreslení pouze goniometrických funkcí sinus a cosinus

7.2.3 Butterfly křivka

Butterfly křivka [3] (Motýlí křivka) je transcendentální křivka daná parametrickými rovnicemi

$$\begin{aligned} x &= sin(t)(e^{cos(t)} - 2cos(4t) - sin^5(\frac{t}{12})) \\ y &= sin(t)(e^{cos(t)} - 2cos(4t) - sin^5(\frac{t}{12})) \end{aligned} \tag{7.3}$$

Zdrojová data jsou obsažena v souboru butterfly.tksl. Oborem hodnot těchto rovnic je křivka ve tvaru „motýlích“ křídel, kterou můžeme vidět na obrázku č.7.4.



Obrázek 7.4: Ukázka Butterfly křivky

Kapitola 8

Závěr

8.1 Shrnutí

Cílem této práce bylo nastínit problematiku řešení elektrických obvodů pomocí diferenciálních rovnic. Bylo předvedeno použití simulačního nástroje TKSL/386 a vysvětlena metoda parazitních indukčností jakožto možné řešení některých problémů se sestavením diferenciálních rovnic a jejich následným výpočtem. Velkým přínosem této části bylo prohloubení znalostí diferenciálních rovnic a to hlavně při řešení elektrických obvodů.

Dalším bodem práce bylo navrhnout grafický systém, který by sloužil jako interaktivní vstupní modul simulačního nástroje TKSL pro práci s grafy. Modul umožňuje práci se vstupními daty programu TKSL/C a mezi jeho základní vlastnosti patří automatické nastavení os, vykreslení mřížky, zobrazení části grafu a také změna definičního oboru a oboru hodnot pro vykreslovanou část grafu. Vývoj tohoto modulu je na počátku a je zde předpoklad pro budoucí přepracování a postupné zdokonalování.V této části jsem získal mnoho praktických zkušeností při objektovém návrhu a implementaci modulu a nelze také přehlédnout získané znalosti z počítačové grafiky.

8.2 Další vývoj

V současném stavu je projekt funkční a je možné jej použít jako modul nebo jej použít jako samostatnou aplikaci, která umí pracovat s výstupním formátem simulačního nástroje TKSL/C. V budoucnu se předpokládá jeho integrace do nově vznikající verze TKSL ve které se předpokládá implementace následujících vlastností:

- zobrazení detailu,
- přepracování grafického vykreslování,
- vykreslování průběhu během výpočtu,
- vylepšení grafického uživatelského rozhraní,
- logaritmické osy,
- export dat grafu pro zpracování jinými aplikacemi (XML,tabulkové editory),

Literatura

[1] Kunovský Jiří. Modern Taylor series method, Technical university of Brno, Faculty of electrical engineering and computer science.
http://www.fit.vutbr.cz/ kunovsky/habil/, 1994.

[2] Kadák Michal. Parazitní kapacity při řešení elektrických obvodů. [bakalářská práce], Vysoké učení technické, Fakulta informačních technologií, 2007.

[3] Wikipedie otevřená encyklopedie. Butterfly curve. [online]
http://en.wikipedia.org/wiki/Butterfly_curve_(transcendental)), Naposledy navštíveno 26. 4. 2008.

[4] Jelen Petr. Grafický systém pro prezentaci výsledků numerických metod. [diplomová práce], Vysoké učení technické, Fakulta elektrotechniky a informatiky, 2001.

[5] PC Svět. .NET. [online] http://www.pcsvet.cz/art/article.php?id=4951, Naposledy navštíveno 24. 4. 2008.

Dodatek A

Uživatelská příručka

Na přiloženém CD je k dispozici jak přeložená verze aplikace, tak i její zdrojové texty. Přeložený grafický systém naleznete v souboru `/binary/TKSLGraphs2.exe`.

A.1 Popis funkcí

Základní programové okno obsahuje menu položku Graph a v ní následující položky.

Load – umožní vybrat zdroj dat ze souboru. Vzorové data jsou na přiloženém CD v adresáři /binary/data.
Exit – ukončí program TKSLGraphs2.

Programové okno grafického systému obahuje menu položku Graph a v ní následující položky.

Export – umožňuje uložit graf jako bitmapu.
Exit – uzavře programové okno grafického systému.

Programové okno grafického systému obsahuje toolbox a v něm následující položky.

Export Graph as Bitmap – umožňuje uložit graf jako bitmapu.
Add Graph – umožní přidání nové křivky z načtených dat.
Edit Graph – umožní změnu parametrů u vybrané přímky v datagridu.
Remove Graph – umožní odstranění vybrané přímky v datagridu.

Programové okno grafického systému obsahuje záložku Axis a v ní následující položky.

Horizontal:
Show Axis – zobrazí/skryje horizontální souřadnou osu.
Show Grid – zobrazí/skryje horizontální část mřížky.
Vertical:
Show Axis – zobrazí/skryje vertikální souřadnou osu.
Show Grid – zobrazí/skryje vertikální část mřížky.

Dodatek B

Obsah přiloženého CD

Obsah přiloženého CD.

`/binary/` – obsahuje přeloženou verzi TKSLGraphs2 spolu se vzorovými daty.

`/doc/` – obsahuje tuto práci ve formátu pdf a komprimovaný soubor thesis.zip se zdrojovými soubrory této práce pro LaTeX.

`/source/` – obsahuje zdrojové kódy grafického systému s vytvořeným projektem do MS Visual Studia 2005.

`/support/` – obsahuje dodatečné programy potřebné pro spuštění TKSLGraphs2.