VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

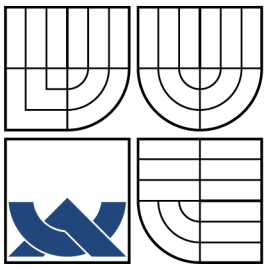
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

HLEDÁNÍ KOŘENŮ POLYNOMU METODOU PŘÍRUSTKU ARGUMENTU

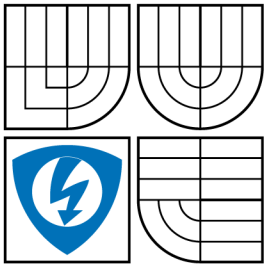
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PAVEL TOŠER
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

HLEDÁNÍ KOŘENŮ POLYNOMU METODOU PŘÍRUSTKU ARGUMENTU

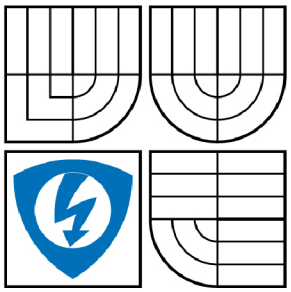
PRINCIPLE OF ARGUMENT INCREMENT FOR SEARCHING POLYNOMIAL ROOTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL TOŠER
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR SADOVSKÝ PH.D.
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student:	Tošer Pavel	***ID:***	77934
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Hledání kořenů polynomu metodou přírůstku argumentu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s metodou přírůstku argumentu a jejím využitím pro hledání kořenů polynomu. V prostředí MATLAB naprogramujte funkci pro hledání násobných kořenů polynomu metodou přírůstku argumentu. Jako lokalizační funkci pro určení kořene polynomu použijte Newtonovu metodu tečen.

DOPORUČENÁ LITERATURA:

[1] ARAMANOVIČ, I.G., LUNC, G.L., ELSGOLC, L.E. Funkcie komplexnej premennej, operátorový počet, teória stability. Bratislava: SNTL/ALFA, 1973.

[2] DOŇAR, B., ZAPLATÍLEK, K. MATLAB - tvorba uživatelských rozhraní. Praha: BEN - technická literatura, 2004.

[3] MÍKA, S. Numerické metody algebry. Praha: SNTL, 1985. ISBN 04-011-85

Termín zadání:	11.2.2008	***Termín odevzdání:***	6.6.2008

Vedoucí práce: Ing. Petr Sadovský, Ph.D.

prof. Dr. Ing. Zbyněk Raida
předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práve třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Pavel Tošer
Bytem:	Onšovice 11, Pelhřimov, 393 01
Narozen/a (datum a místo):	9. listopadu 1985 v Pelhřimově

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

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

   ☐ disertační práce
   ☐ diplomová práce
   ☒ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

Název VŠKP:	Hledání kořenů polynomů metodou přírustku argumentu
Vedoucí/ školitel VŠKP:	Ing. Petr Sadovský Ph.D.
Ústav:	Ústav fyziky
Datum obhajoby VŠKP:	_____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

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.

* hodící se zaškrtněte

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 pořizovaní 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: 6. června 2008

…….. Nabyvatel

…… Autor

ABSTRAKT

Existuje celá řada metod, které se požívají k nalezení kořenů polynomů. Ve většině případů jde o metody, které se používají pouze na speciální případy řešení. Tato práce se zabývá vývojem metody, která by byla schopna efektivně pracovat i v případě, že polynom má vícenásobné kořeny. Postup spočívá ve vyběru vhodné iterační metody v kombinaci s metodou přírustku argumentu. Doposud není znám algoritmus, který by řešil tuto úlohu tímto způsobem. Proto by měl předložený postup odstranit nedostatky již existujících metod a doplnit je o nové poznatky.

KLÍČOVÁ SLOVA

Metoda přírustku argumentu, Newtonova metoda tečen, násobné kořeny polynomů, stabilita systému, nuly a póly.

ABSTRACT

Several methods exist for searching multinominal roots. Methods in more cases are used only for special solves. The goal of this thesis is to discover solution for searching multinominal roots. The process is based on optimal iterative method in combination with priciple of argument increment. There is no procedure solving it in this way up to now. This method removes shortcomings already existing methods and she could also complement them with a new knowledge.

KEYWORDS

Principle of argument increment, Newton method of secant, polynomial roots, multinominal roots, system stability, zeros and poles.

TOŠER, P. *Hledání kořenů polynomu metodou přírůstku argumentu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 53 s. Vedoucí bakalářské práce Ing. Petr Sadovský, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma *Hledání kořenů polynomů metodou přírustku argumentu* jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Petru Sadovskému, Ph.D. za pedagogickou, účinnou metodickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

1 ÚVOD

Pro hledání kořenů matematických funkcí existuje řada známých metod. Jednotlivé metody se liší výpočetní náročností, tedy časovou náročností, dále pak rychlostí konvergence, nebo oborem konvergence. Mezi známé iterační metody patří například metoda sečen, metoda půlení intervalu, metoda regula falsi a také metoda tečen. Hlavní vlastností těchto metod je, že s jejich pomocí je hledán bod na dané křivce, který má nulovou funkční hodnotu. Všechny tyto uvedené metody se používají při řešení nelineárních rovnic, které nelze řešit jednoduchým způsobem.

Tato práce si klade za cíl navrhnout novou metodu pro hledání kořenů polynomů, která bude určena k hledání především násobných kořenů polynomů, protože ostatní uvedené metody u násobných kořenů více či méně selhávají. Dosud není znám algoritmus, který by řešil danou úlohu tímto způsobem, proto je namístě metodu přírustku argumentu vyzkoušet a porovnat její dosažené výsledky s výsledky již existujících postupů.

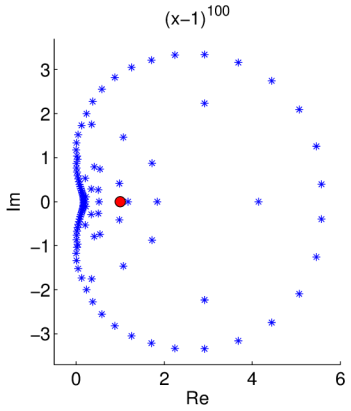
Jako pracovní prostředí bylo vybráno prostředí Matlab. Program Matlab je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích nebo komunikačních systémů. Je to nástroj, jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací. Výhoda spočívá ve velkých možnostech tohoto prostředí, neboť je velmi rozšířeno v průmyslu, vědě a v neposlední řadě jeho verze existují pro řadu operačních systémů.

Výsledky práce mohou přinést nové poznatky do oborů jako je například matematika a teoretická elektrotechnika. Souvislost s elektrotechnikou se odvíjí od vlastností spojitých dynamických systémů, které jsou popsány několika způsoby. Mezi něž patří například frekvenční, přechodová, impulsní charakteristika nebo rozložení nul a pólů přenosových funkcí. Díky vztahům je možné přecházet mezi jednotlivými popisy podle daných pravidel. Protože práce se zaměřuje na hledání kořenů polynomů, a ty mají své opodstatnění ve spojitých dynamických systémech (rozložení nul a pólů přenosu), mohou být dosažené výsledky dále využity pro další výpočty, nebo případně použity k vyhodnocování stability spojitých dynamických systémů. Dalším důvodem pro realizaci funkcí jsou nepřesvědčivé výsledky v oblasti lokalizace násobných kořenů, které podává programové prostředí Matlab. Využitím nové metody by měly být takové nedostatky redukovány, nebo zcela odstraněny.

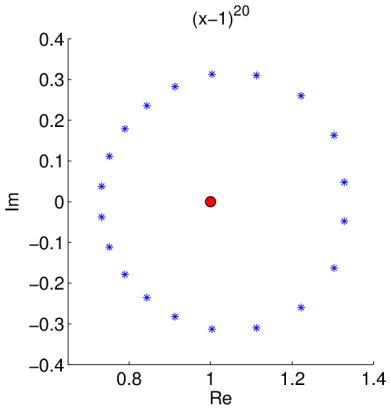
Součástí práce je také vytvoření uživatelského prostředí v Matlabu s implementovanými nově navrženými funkcemi. Dosažené výsledky by pak měly být porovnány s výsledky existujících algoritmů přímo v navržené aplikaci.

2 ROZBOR PROBLEMATIKY

Mezi hlavní důvody aplikace metody přírustku argumentu patří ověření výpočetních možností nové metody na různých reálných polynomech. Na vykreslených obrázcích je vidět, jakým způsobem pracuje při výpočtech násobných kořenů polynomů funkce implementovaná v programovém prostředí Matlab.



Obrázek 2.1: Rozdílná lokalizace pro 100 násobný kořen.



Obrázek 2.2: Rozdílná lokalizace pro 20-ti násobný kořen.

Je zřejmé, že po vykreslení kořenů polynomu $(x-1)^{100}=0$, nebyl nalezen jeden jedinný stonásobný kořen, ale byl lokalizován shluk blízkých kořenů. Měl by však být vykreslen pouze jeden kořen, jak naznačují obrázky 2.1, 2.2. Správné řešení reprezentuje bod $1+0i$. Shluk blízkých kořenů postupně narustá s zvyšující se násobností kořene, a tím dochází ke stále rostoucím nepřesnostem. Zvláštností takto nalezených kořenů je ten fakt, že součet všech reálných a imaginárních částí kořenů dělený celkovým počtem kořenů dává správný výsledek 2.2. Z takového výsledku však nelze určit celkovou násobnost kořene. Podle obrázků je také velice dobře patrné, jak má každý komplexní kořen svůj komplexně sdružený protějšek.

Například pro polynom zadaný tímto zápisem

$$x^5-5x^4+10x^3-10x^2+5x-1, \tag{2.1}$$

by měl být správným výsledkem pětinásobný kořen

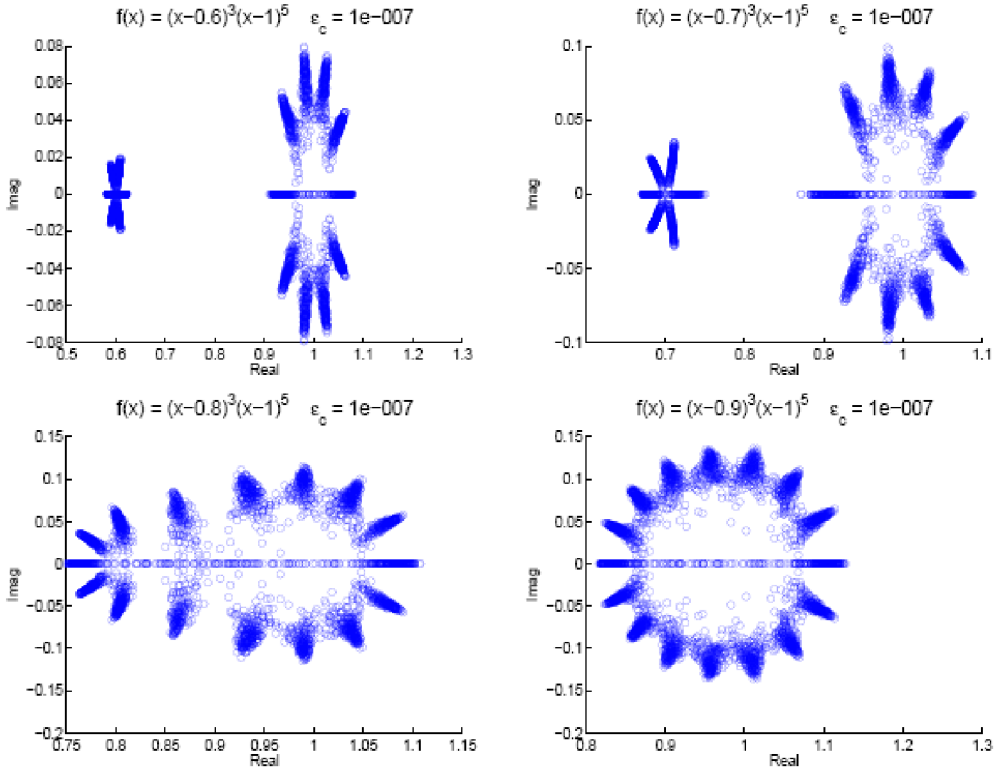
$$x_{1,2,3,4,5}=1. \tag{2.2}$$

Pro srovnání funkce implementovaná v prostředí Matlab, která využívá maticového rozkladu polynomu a řešení N rovnic o N neznamých dosahuje pro polynom (2.1) těchto výsledků:

$$\begin{aligned} x_1 &= 1.0010 + 0.0007i, \\ x_2 &= 1.0010 - 0.0007i, \\ x_3 &= 0.9996 + 0.0012i, \\ x_4 &= 0.9996 - 0.0012i, \\ x_5 &= 0.9987. \end{aligned} \tag{2.3}$$

Srovnání s předchozím výsledkem (2.2) je zřejmé, že nalezené kořeny jsou odlišné.

Na dalším obrázku 2.3 je znázorněno rozložení pravděpodobností nalezení kořenů čtyř podobných polynomů. Tyto polynomy jsou specifické tím, že polynom je tvořen rozdílnou dvojicí násobných kořenů, které se čím dál více svou hodnotou přibližují k druhému kořeni (viz. popis jednotlivých obrázků). Je vidět jak se postupným přibližováním křivky více podobají zmíněnému shluku kořenů.



Obrázek 2.3: Distribuce kořenů čtyř podobných polynomů.

Otázkou je, jakou iterační metodu využít k lokalizaci kořenů polynomu. Protože každá z iteračních metod využívá specifického způsobu řešení, nehodí se všechny pro řešení daného problému.

Rozdělení iteračních metod

- jednobodové - metoda půlení intervalu (která je v jistém smyslu univerzální jednobodovou metodou), Newtonova metoda,
- dvoubodové - např. bisekce, regula falsi, metoda sečen,
- vícebodové.

Dělení z programátorského hlediska

- nevyžadující derivaci - meoda bisekce, regula falsi, metoda sečen a obvykle MPI (záleží na zvoleném iteračním vzorci),
- vyžadující znalost první derivace - Newtonova metoda,
- vyžadující znalost vyšších derivací.

Dělení metod rešení rovnic podle konvergence

- vždy konvergentní - metoda bisekce a regula falsi,
- ostatní - Newtonova, metoda sečen, MPI.

Uvedené metody mohou být dále rozděleny podle toho, zda jsou schopny vyhledávat násobné kořeny, nebo zda mohou pracovat v komplexním oboru řešení rovnic.

Hledání násobných kořenů

V okolí kořene sudé násobnosti funkce nemění znaménko, takže nelze použít metody bisekce a regula falsi. Metoda sečen a Newtonova metoda jsou sice použitelné pro hledání násobných kořenu, ale jejich konvergence je pak prvního řádu. U metodě prosté iterace záleží pouze na použitém iteračním vzorci, nikoliv na násobnosti kořene původní rovnice [10].

Rešení rovnic v komplexním oboru

Metoda bisekce a metoda regula falsi jsou závislé na úplném uspořádání reálných čísel. Metoda sečen a Newtonova metoda jsou použitelné pro komplexní kořeny. Pro nalezení komplexních kořenů muže být nutný počáteční odhad s nenulovou imaginární částí [10].

Díky všem předešlým vlastnostem byla vybrána za lokalizační Newtonova metoda tečen, mezi jejíž přednosti patří hlavně velice rychlá konvergence a vlastnost, že se jedná o jednobodovou metodu. Naopak mezi nedostatky patří nutnost výpočtu derivace funkce a ne vždy stoprocentní konvergence.

3 VLASTNOSTI POLYNOMŮ

Polynomy jsou jedny z nejjednodušších funkcí, proto se také v řadě technických problémů nahrazují velice složité funkce právě polynomy. Matematické vlastnosti polynomu například dobře vyjadřují vlastnosti skutečného výsledného systému (např. dynamika, rychlost, stabilita, vibrace, tlumení, citlivost na změny nebo poruchy).

3.1 Definice polynomu

Polynomem je funkce f definovaná v reálném oboru předpisem:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0, \tag{3.1}$$

kde $a_0, a_1, ..., a_n$ jsou reálná čísla, $a_n \neq 0$. Číslo n se nazývá *stupeň polynomu*. Pro polynom n-tého stupně se využívá obvykle označení P_n. Polynom stupně 0, tedy funkce f definovaná na R předpisem [9]

$$f(x) = c, \tag{3.2}$$

kde c je reálné číslo, se nazývá *konstanta*. Je-li funkční hodnota polynomu v čísle x_0 rovna nule, a tedy platí [9]

$$a_n x_0^n + a_{n-1} x_0^{n-1} + ... + a_1 x_0 + a_0 = 0, \tag{3.3}$$

nazývá se číslo x_0 *kořenem polynomu*.

Některé důležité vlastnosti polynomu a jejich kořenů:

- Základní věta algebry:

Každý polynom stupně $n \geq 1$ má alespoň jeden kořen [9].

- Věta Bézoutova:

Číslo x_0 je kořenem polynomu P_n stupně $n \geq 1$, právě když platí [9]

$$P_n(x) = (x - x_0) Q_{n-1}(x), \tag{3.4}$$

kde Q_{n-1} je vhodný polynom stupně $n-1$. Výraz $(x - x_0)$ vystupující v předchozím vztahu (3.4) se nazývá *kořenový činitel* příslušný ke kořenu x_0 [9].

- Rozklad polynomu na kořenové činitele:

Jsou-li (reálná nebo komplexní, ne nutně různá) čísla $x_1, x_2, ..., x_n$ kořeny polynomu $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$, platí [9]

$$P_n(x) = a_n(x - x_1)(x - x_2)....(x - x_n). \tag{3.5}$$

Odtud plyne, že polynom stupně n má právě n (ne nutně různých) kořenů. Mezi koeficienty polynomu a jeho kořeny platí vztahy zvané Vietovy vzorce [9].
Je-li

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 = a_n(x - x_1)(x - x_2)....(x - x_n),$$

platí:

$$\begin{aligned} a_{n-1} &= -a_n(x_1 + x_2 + ... + x_n), \\ a_{n-2} &= a_n(x_1x_2 + x_1x_3 + ... + x_2x_3 + ... + x_{n-1}x_n), \\ a_0 &= (-1)^n a_n(x_1x_2...x_n). \end{aligned} \tag{3.6}$$

Vlastnosti kořenů algebraické rovnice s reálnými koeficienty

1. Má-li algebraická rovnice s reálnými koeficienty komplexní kořen $\alpha = a + bi$, má také kořen $\alpha = a - bi$ (číslo komplexně sdružené k α).

2. Pokud má algebraická rovnice s reálnými koeficienty vícenásobný komplexní kořen, potom číslo komplexně sdružené je také vícenásobným kořenem této rovnice a násobnosti obou kořenu jsou stejné.

3. V případě, že má algebraická rovnice s reálnými koeficienty komplexní kořeny, je jejich počet sudý.

4. Každá algebraická rovnice s reálnými koeficienty lichého stupně má alespoň jeden reálný kořen [9].

3.2 Hornerovo schéma

Existuje také jeden velmi jednoduchý algoritmus nazvaný Hornerovo schéma, s jehož pomocí je možné vypočítat funkční hodnotu polynomu v daném bodu. P je polynomem a $x_0 \in R$.
Existují polynomy Q, R tak, že platí [9]

$$P(x) = (x - x_0)Q(x) + R(x), \tag{3.7}$$

kde stupeň $R <$ stupeň $(x - x_0)$, tedy je roven nule a R je konstanta [9]. Po dosazení x_0 do předchozí rovnosti je výsledkem

$$P(x_0) = R, \text{ tedy } P(x) = (x - x_0)Q(x) + P(x_0). \tag{3.8}$$

Nechť tedy

$$P(x) = \sum_{i=0}^{n} a_i x^i aQ(x) = \sum_{i=0}^{n-1} b_i x^i \tag{3.9}$$

Potom platí [9]

$$\begin{aligned} &\sum_{i=0}^{n} a_i x^i = (x - x_0) \sum_{i=0}^{n-1} b_i x^i + P(x_0) = b_{n-1}x^n + \cdots \\ &\cdots + \sum_{i=0}^{n-1} (b_{i-1} - b_i x_0)x^i + P(x_0) - b_0 x_0. \end{aligned} \tag{3.10}$$

Porovnáním koeficientů jsou výsledkem rovnosti uvedené v levé části následující tabulky, zatímco v pravém sloupci jsou rovnosti z nich jednoduše odvozené. V pravém sloupci je naznačen výpočet koeficientů částecného podílu Q včetně hodnoty $P(x_0)$ polynomu P v bodě x_0.

$a_n b_{n-1}$	$b_{n-1} = a_n$
$a_{n-1} = b_{n-2} - b_{n-1}x_0$	$b_{n-2} = a_{n-1} + x_0 b_{n-1}$
.	.
.	.
.	.
$a_i = b_{i-1} - b_i x_0$	$b_{i-1} = a_i + x_0 b_i$
.	.
.	.
.	.
$a_1 = b_0 - b_1 x_0$	$b_0 = a_1 + x_0 b_1$
$a_0 = P(x_0) - b_0 x_0$	$P(x_0) = a_0 + x_0 b_0.$

Tento postup se zpravidla zapisuje ve tvaru následující tabulky, kde x_0 je kořen polynomu P [9].

a_n	a_{n-1}	...	a_i	...	a_1	a_0
	$x_0 b_{n-1}$	...	$x_0 b_i$	...	$x_0 b_1$	$x_0 b_0$
b_{n-1}	b_{n-2}	...	b_{i-1}	...	b_0	$p(x_0)$

4 PRINCIPY ITERAČNÍCH METOD

Pro každou iterační metodu jsou důležitá dvě hlediska. Prvním hlediskem je zda konverguje posloupnost iterací k hledanému kořeni, a druhým pokud konverguje, pak jak rychle? Jestliže před započetím výpočtu nejsou známy potřebné informace a je znám například pouze výpočetní interval, je výhodné použití iteračních metod. Jejich konvergence závísí na volbě počátečních aproximací. Vždy konvergentní metody mají tu nevýhodu, že konvergují pomalu, proto se hodí pro určení počáteční aproximace rychleji konvergující metody. Z těchto vět je patrné, že rychlost konvergence velmi záleží na vhodně zvoleném startovacím bodu. Metody řešení díky těmto úvahám lze rozdělit na tyto části:

1. Startovací metody (metody vždy konvergentní)
2. Zpřesňující metody
3. Speciální metody (pro polynomy)

4.1 Newtonova metoda tečen

Tato metoda může být zařazena mezi metody zpřesňující. Už sám název metody říká, že se pracuje s tečnami ke grafu funkce f. Newtonova metoda tečen předpokládá, že funkce f má derivaci. Pokud tato metoda koverguje, konverguje velice rychle. Je to metoda druhého řádu, počet platných desetinných míst výsledky se v každé iteraci zdvojnásobuje.

Newtonova metoda se popisuje graficky takto: po zvolení počátecní aproximace kořene x_0 je bodem $[x_0, f(x_0)]$ vedena tečna ke grafu funkce f. Její průsečík s osou x se označí x_1. Následně je vedena tečna bodem $[x_1, f(x_1)]$, její průsečík s osou x je x_2 atd.

Průsečík tečny v bodě $[x_k, f(x_k)]$ s osou x se vypočítá jako [10]

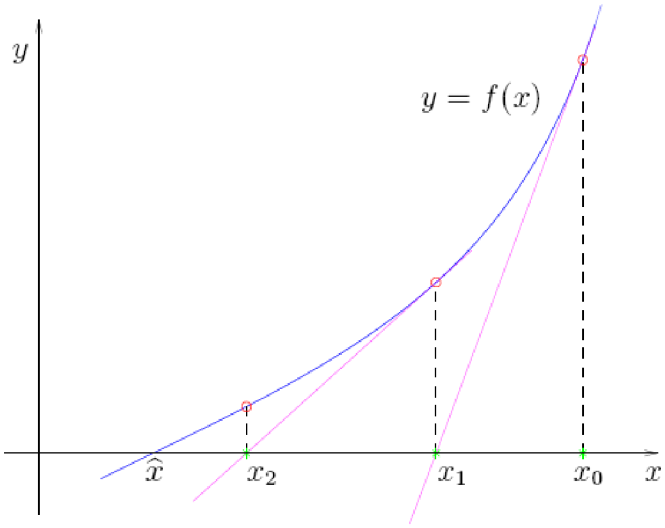
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{4.1}$$

Výpočet trvá tak dlouho, dokud není splněna podmínka

$$|x_k - x_{k-1}| < \epsilon. \tag{4.2}$$

Při splnění této podmínky však nemusí platit $|x_k - \varepsilon| < \epsilon$. [1]

[1] Pod pojmem přesnost řešení je brán v daném případě vždy absolutní rozdíl mezi dvěmi následujícími vypočítanými hodnotami. To znamená, že se hledá řešení, při kterém další výpočet již nevede k podstatnému zlepšení výsledku.



Obrázek 4.1: Princip Newtonovy metody tečen.

Newtonovu metodu lze odvodit i pomocí Taylorova vzorce. Při předpokladu, že je známa k-tá aproximace rešení x_k. Pak lze psát [10]

$$f(\varepsilon) = f(x_k) + f^{'}(x_k)(\varepsilon - x_k) + R, \tag{4.3}$$

kde R je zbytek v Taylorove vzorci.
Zanedbáním tohoto zbytku a následnou úvahou: $f(\varepsilon) = 0$ (protože ε je kořenem rovnice $f(x) = 0$), je z předchozí rovnice přibližně vyjádřen kořen ε jako

$$\varepsilon \doteq x_k - \frac{f(x_k)}{f'(x_k)}, \tag{4.4}$$

což je právě x_{k+1} nalezené dříve popsaným způsobem.
Z Taylorova vzorce lze také odvodit odhady chyby k-té aproximace kořene získané Newtonovou metodou. Má-li funkce na intervalu I obsahujícím x_k i kořen ε druhou derivaci, platí [10]

$$|\varepsilon - x_k| \leq \frac{M_2}{2m_1}(x_k - x_{k-1})^2, \tag{4.5}$$

$$|\varepsilon - x_k| \leq \frac{M_2}{2m_1}(\varepsilon - x_{k-1})^2, \tag{4.6}$$

kde $M_2 = max|f^{''}(x)|$ a $m_1 = min|f^{'}(x)|$ pro $x \in I$.
Newtonova metoda je z metod pro rešení nelineárních rovnice nejefektivnější, nemusí však vždy konvergovat. Jestli Newtonova metoda konvergovat bude, nebo nebude, závisí do značné míry také na tom, jak je zvolena počáteční aproximaci x_0 [10].

Základní nevýhody jsou:

1. Zadaná funkce f musí být diferencovatelná.
2. Derivace se přímo vyskytuje v iterační formuli.
3. V každé iteraci se musí kromě funkční hodnoty počítat také hodnota derivace.

4.2 Modifikovaná Newtonova metoda tečen

Protože obyčejně není násobnost kořene předem známá, existuje způsob jak upravit Newtonovu metodu tečen. Modifikace spočívá v úpravě iteračního vzorce, což má za následek zrychlení konvergence, a tudíž snížení počtu potřebných iterací, než je dosaženo požadované přesnosti.
Metoda je založena na poznatku, že funkce:

$$u(x_i) = \frac{f(x_i)}{f'(x_i)} \tag{4.7}$$

má v bodě $x = \alpha$ jednoduchý kořen, bez ohledu na násobnost kořene původní funkce. Uvažujeme-li místo rovnice $f(x) = 0$ rovnici $u(x) = 0$, jsou kořeny této rovnice totožné s kořeny předchozí rovnice a jsou všechny jednoduché. K tomu aby byla vytvořena nová iterační metoda, jejíž řád konvergence je nezávislý na násobnosti kořene, pak stačí pouze zaměnit obě uvedené funkce. Po takovéto záměně přejde Newtonova-Raphsonova metoda po úpravě na následující vzorec [1].

$$x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)} \tag{4.8}$$

Efektivnost metody je však nižší než u původní, neboť je při jejím užití nutné vypočítat o jednu derivaci více než původně. Pokud je však násobnost kořene známa, lze speciálně upravit Newtonovu-Raphsonovu metodu tak, aby konvergovala kvadraticky i pro r-násobný kořen. Toho lze dosáhnout takovýmto zápisem [1] :

$$x_{i+1} = x_i - r\frac{u(x_i)}{u'(x_i)} \tag{4.9}$$

Protože při každém kroku je nutné vypočítat funkční hodnoty $f(x)$ a $f'(x)$ v bodu x_i, může být náročný výpočet derivace. Pokud se tato derivace nijak významně nemění, lze využít tohoto vztahu:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_0)} = x_i - \frac{f(x_i)}{c} \tag{4.10}$$

kde $c = f'(x_0)$, $n = 0, 1 \ldots$
V případě, že $f(x)$ je polynom je ušetřena téměř polovina operací.

Vlivy na modifikaci:

- Dochází ke zrychlení konvergence
- Nutné prověření podmínek konvergence

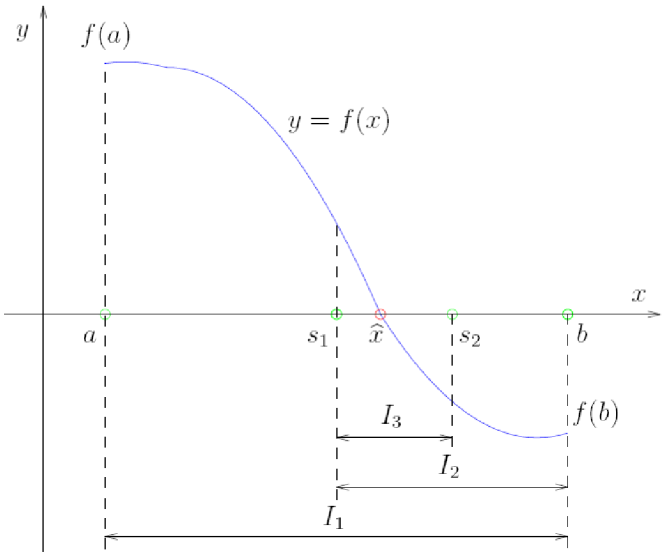
4.3 Metoda půlení intervalu

Metoda pulení intervalu je nejjednodušší starovací metodou rešení nelineárních rovnic. Je známa i pod názvem *bisekce*. Vstupem je interval $< a, b >$ takový, že $f(a)\Delta f(b) < 0$, tj. leží v něm alespoň jeden kořen rovnice $f(x) = 0$. Tento výchozí interval je označen jako $< a_0, b_0 >$.
Dalším krokem je rozpůlení intervalu. Jeho střed je dán tímto vztahem [10]

$$x_0 = \frac{a_0 + b_0}{2} \tag{4.11}$$

Z intervalů $< a_0, x_0 >$, $< x_0, b_0 >$ musí být vybrán ten, ve kterém je zaručena existence kořene. Který z nich to je, lze rozeznat podle znamének funkčních hodnot v krajních bodech. Je-li $f(a_0)\Delta f(x_0) < 0$, výpočet pokračuje s intervalem $< a_0, x_0 >$, v opačném případě s intervalem $< x_0, b_0 >$. Platí-li $f(x0) = 0$, pak je nalezen kořen rovnice a výpočet může být ukončen.

Nový interval poloviční délky má označení $< a_1, b_1 >$, opět je nutné jej rozpůlit a stejným způsobem pokračovat s výpočtem [10].



Obrázek 4.2: Princip metody půlení intervalu.

Tímto postupem jsou sestrojeny posloupnosti intervalů $< a_0, b_0 >$, $< a_1, b_1 >$, $< a_2, b_2 >$. Každý další interval bude získán tak, že z předchozího (na základě

znamének funkčních hodnot v krajních bodech a uprostřed) bude vybrána ta polovinu, která obsahuje kořen rovnice.

V půlení algoritmus pokračuje tak dlouho, dokud není nalezen kořen rovnice, nebo dokud se interval nezúží na předem danou délku 2ϵ neboli dokud pro nějaké k neplatí [10]

$$b_k - a_k < 2\epsilon \tag{4.12}$$

Za přibližnou hodnotu kořene je pak brán střed posledního nalezeného intervalu.

$$x_k = \frac{a_k + b_k}{2} \tag{4.13}$$

Protože kořen se určitě nachází uvnitř posledního intervalu, může se x_k od přesné hodnoty kořene lišit nanejvýš o polovinu jeho délky, tj. o ϵ,

$$|x_k - \varepsilon| < \epsilon. \tag{4.14}$$

Touto metodou je kořen rovnice naleznen vždy. V případě, že na výchozím intervalu $< a, b >$ je více kořenů, metodou půlení intervalu je nalezen jeden z nich. Výhodou je kromě její jednoduchosti i fakt, že se dá předem určit počet kroků, potřebných k dosažení požadované přesnosti (lze ukázat, že pro zpřesnění o jedno desetinné místo je třeba provést přibližně 3,3 iterace) [10].

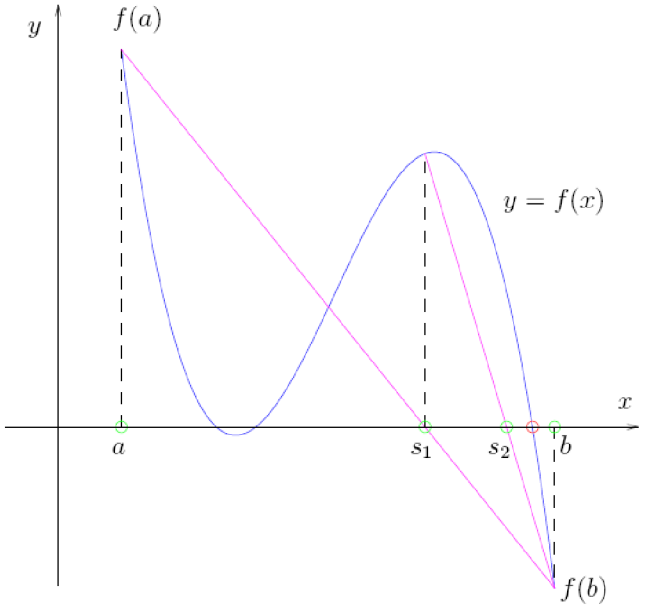
Nevýhodou kromě pomalé konvergence je fakt, že se tato metoda nedá použít pro určení komplexního kořene. Dále metoda půlení intervalu konverguje dosti pomalu. Proto je vhodné použít ji na zúžení původního intervalu a pak pokračovat jinou, rychlejší metodou.

4.4 Metoda regula falsi

Princip metody regula falsi je velmi podobný jako u metody pulení intervalu. Opět je postupně zúžován interval obsahující kořen rovnice. Tentokrát dělicím bodem není polovina intervalu, ale prusečík sečny vedené body $[a_k, f(a_k)]$ a $[b_k, f(b_k)]$ s osou x. Tento prusečík lze vypočítat podle vzorce [10]

$$x_k = b_k - \frac{b_k - a_k}{f(b_k) - f(a_k)} f(b_k) \tag{4.15}$$

Z intervalů $< a_k, x_k >$, $< x_k, b_k >$ se vybere ten, v jehož krajních bodech mají funkční hodnoty funkce f opačná znaménka.
Platí-li $f(a_k) \cdot f(x_k) < 0$, položíme $a_{k+1} = a_k, b_{k+1} = x_k$, platí-li $f(b_k) \cdot f(x_k) < 0$, položíme $a_{k+1} = x_k, b_{k+1} = b_k$. V případě, že $f(x_k) = 0$, byl nalezen kořen rovnice a výpočet může skončit.



Obrázek 4.3: Princip metody Regula falsi.

Výpočet pokračuje tak dlouho, dokud není nalezen kořen, nebo dokud neplatí [10]

$$|x_k - x_{k-1}| < \epsilon \tag{4.16}$$

kde $\epsilon > 0$ je předem dané číslo. Splněním tohoto kriteria ale bohužel není zaručeno, že přesná hodnota kořene ϵ se od jeho aproximace x_k liší o méne než ϵ. Pro ověření výsledku platí $|x_k - \epsilon| < \epsilon$, následuje $f(x_k + \epsilon)$ a $f(x_k - \epsilon)$. Platí-li $f(x_k) \cdot f(x_k + \epsilon) < 0$, resp. $< f(x_k) \cdot f(x_k - \epsilon) < 0$, je jisté, že kořen ε leží v intervalu $< x_k, x_k + \epsilon >$, resp. $< x_k - \epsilon, x_k >$, a tedy se od x_k nemůže lišit o více než ϵ [10].

Vlastnosti metody jsou následující:

Křivka se nahradí v daném intervalu přímkou. Průsečík této přímky s osou x je zpřesnění kořene. Metoda regula falsi je konvergentní pro všechny spojité funkce. Obecně se jedná o nestacionární metodu, mnohdy (např. u konvexních funkcí, kdy je druhá derivace kladná) je stacionární - jeden krajní bod intervalu je vždy jedním ze dvou bodů užitých v následující iteraci.

Tato metoda je velmi vhodná v případě, že nejsou známy výchozí údaje o poloze kořenu, konverguje však relativně pomalu. Stačí ale nalézt pouze dva body, ve kterých má hodnota funkce opačné znaménko a pak tuto metodu lze aplikovat na zadaný interval.

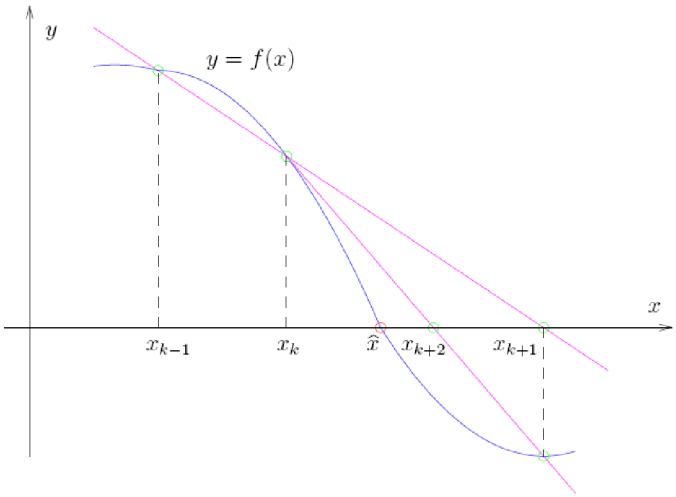
4.5 Metoda sečen

Metoda sečen je velmi podobná metodě regula falsi. Pro zahájení výpočtu je potřeba znát dvě počáteční aproximace, ale na rozdíl od Newtonovy metody je počítána v každém kroku pouze jedna nová funkční hodnota, což je úspora času.

Vychází se z intervalu $< a, b >$ obsahujícího kořen rovnice. Po označení $x_0 = a$ a $x_1 = b$, je dalším krokem vedení sečny body $[x_0, f(x_0)]$ a $[x_1, f(x_1)]$. Takto je nalezen její prusečík s osou x (jeho označení x_2).
Narozdíl od metody regula falsi není vybírán interval obsahující kořen, ale je vedena sečna body $[x_1, f(x_1)]$, $[x_2, f(x_2)]$, její průsečík má označení x_3, pak je vedena sečna body $[x_2, f(x_2)]$ a $[x_3, f(x_3)]$ atd.
V k-tém kroku metody je počítána aproximace kořene podle vzorce [10]

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k), \tag{4.17}$$

kde $x_0 = a$, $x_1 = b$.



Obrázek 4.4: Princip metody sečen.

Výpočet končí, když je splněna podmínka [10]

$$|x_k - x_{k-1}| < \epsilon \tag{4.18}$$

nebo když je výpočtem nalezen přímo kořen rovnice. Daná podmínka nezaručuje, že platí $|x_k - \varepsilon| < \epsilon$.

Metoda sečen je rychlejší než metoda regula falsi, nemusí ale vždy konvergovat. Protože je obtížné předem zjistit, zda metoda pro danou rovnici konverguje nebo diverguje, je vhodné zadat při výpočtu maximální počet kroků. Je-li tento počet překročen a kořen rovnice není nalezen, výpočet končí s tím, že metoda diverguje. Pak je nutno změnit počáteční aproximace, nebo zvolit jinou metodu.

5 PRINCIP PŘÍRUSTKU ARGUMENTU

Tato kapitola se zabývá problematikou, jakým způsobem lze určit počet kořenů nacházejících se uvnitř libovolné souvislé oblasti včetně její hranice. Tato myšlenka je známá z teorií funkcí komplexních proměnných pod názvem Cauchyho princip argumentu [4]. Metoda byla později využita Harry Nyquistem pro určování podmínek stability.

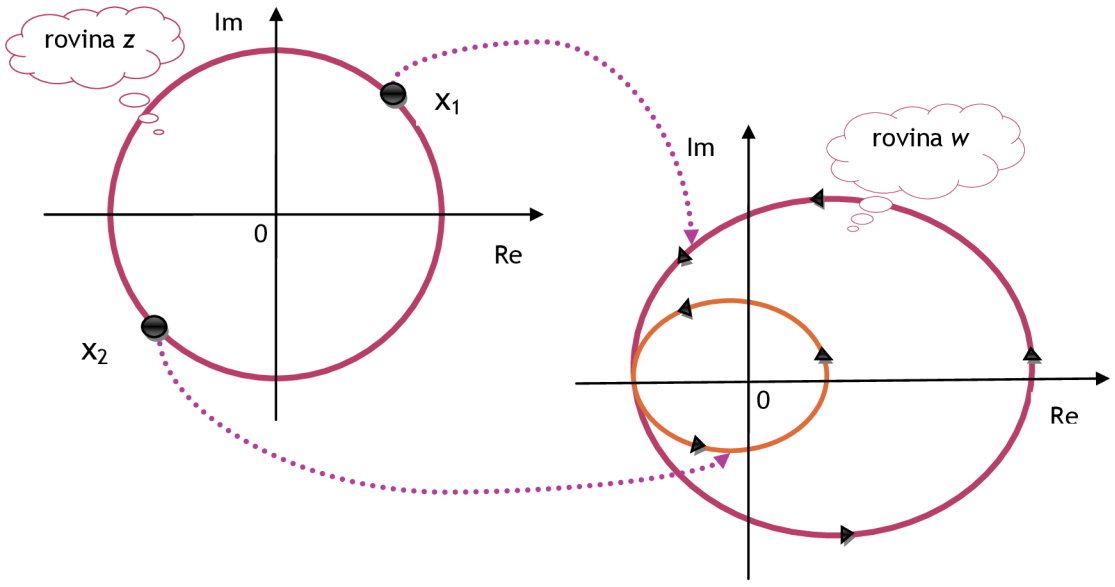
Za předpokladu, že komplexní funkce $\omega = F(z)$ je analytická v jednoduše souvislé oblasti D včetně její hranice C a že nemá žádné nulové body ležící na této hranici, potom je počet kořenů N rovnice

$$F(z) = 0 \tag{5.1}$$

ležících uvnitř oblasti D roven absolutní hodnotě přírustku argumentu na křivce vytvořené pomocí funkce $\omega = F(z)$ dělenému veličinou 2π, když proměnná z ve své rovině oběhne hranici C. Platí tedy [4], [5]:

$$N = \frac{1}{2\pi i}\int_C \frac{f'(z)}{f(z)}dz = \frac{1}{2\pi}|\Delta_c Arg f(z)| \tag{5.2}$$

počet kořenů rovnice ležících uvnitř křivky C je roven počtu otáček kolem počátku souřadnic vektoru $\omega = F(z)$ v komplexní rovině ω, když proměnná z jednou oběhne hranici C. Tyto definice jsou uvedeny [5][2][4].



Obrázek 5.1: Princip metody přírustku argumentu.

5.1 Matematický důkaz přírustku argumentu

Platnost vztahu (5.2) dokládají následující věty:

Věta 1. Nechť $f(z)$ je funkce holomorfní na mezikruží daném těmito parametry $P(z_0, r, R)$, kde $0 \leq r \leq R \leq \infty$. Pak existují koeficienty $a_n \in C$ (obor komplexních čísel), $n \in Z$ (obor celých čísel), že [14]

$$f(z) = \sum_{n=-\infty}^{\infty} a_n (z - z_0)^n \tag{5.3}$$

Funkce je meromorfní, pokud je holomorfní na prstencovém okolí $U(z_0, \varepsilon)/\{z_0\}$, přičemž z_0 není v definičním oboru této funkce. Bod z_0 se nazývá izolovaným singulárním bodem (singularitou) této funkce. Singularita může být odstranitelná, pokud $\lim_{z \to z_0}(f)$ je vlastní číslo, nebo se nazývá pól funkce, pokud $\lim_{z \to z_0}(f) = \infty$ nebo se jedná, pokud $\lim_{z \to z_0}(f)$ neexistuje, o podstatnou singularitu.

Věta 2. Nechť $C \subset D$, D je jednoduše souvislá oblast, C je jednoduše uzavřená křivka, f je analytická funkce. Nechť $f(z_0 = 0)$. Pak

$$\frac{1}{2\pi i} \int_C \frac{f'(z)}{f(z)} dz = N, \tag{5.4}$$

kde C je křivka, v jejíž vnitřní oblasti leží bod z_0 a N je násobnost kořene z_0 [14].

Věta 3. Nechť funkce f je holomorfní na $\varepsilon-$ ovém okolí $U(z_0, \varepsilon)$ bodu z_0. Pak bod z_0 je k-násobným kořenem funkce f právě tehdy, když

$$f(z) = (z - z_0)^k g(z), \tag{5.5}$$

kde $g(z)$ je holomorfní funkce na $U(z_0, \epsilon)$ a $g(z_0 \neq 0)$[14].

Násobnost nulového bodu

Postačí dokázat, že k je násobnost kořene z_0 a $g(x)$ je holomorfní funkce. Podle Věty 1. je možné funkci f (funkce f je v předpokladu Věty 3 *holomorfní*) zapsat jako Laurentovu řadu [14]

$$f(z) = a_0 + a_1(z - z_0) + a_2(z - z_0)^2 + \cdots \tag{5.6}$$

Při úvaze, že má funkce kořen v z_0, pak je několik prvních koeficientů a_n této řady nulových. Následně při předpokladu, že první nenulový koeficient této řady je a_k. Funkce má potom rozvoj

$$\begin{aligned} f(z) &= 0 + 0(z - z_0) + \cdots + 0(z - z_0)^{k-1} + a_k(z - z_0)^k + \cdots \\ &+ a_{k+1}(z - z_0)^{k+1} + a_{k+2}(z - z_0)^{k+2} + \cdots \end{aligned} \tag{5.7}$$

Vytknutím členu $(z - z_0)^k$ z této řady je výsledkem

$$f(z) = (z - z_0)^k \left[a_k + a_{k+1}(z - z_0) + a_{k+2}(z - z_0)^2 + \cdots\right] \tag{5.8}$$

$$f(z) = (z - z_0)^k g(z) \tag{5.9}$$

Jak je zřejmé, funkce $g(z)$ je součet mocninné řady s otevřeným koncem. Jedná se tedy o funkci *holomorfní* (viz. Věta 1). Funkce $f(z)$ a $g(z)$ mají společné kořeny až na kořen funkce $f(z)$, jehož násobnost je hledána [14], [4].

Koeficienty $a_k, a_{k+1}, a_{k+2}, a_{k+3}, \cdots$ jsou nenulové, takže platí $g(z_0) = a_k \neq 0$ z čehož vyplývá také $g(z) \neq 0$ na okolí z_0.
Pro derivaci funkce $f(z)$ platí

$$f'(z) = \left[(z - z_0)^k \cdot g(z)\right]' = k(z - z_0)^{k-1} g(z) + (z - z_0)^k g'(z) \tag{5.10}$$

Vyjádření podílu $f'(z)$ a $f(z)$

$$\frac{f'(z)}{f(z)} = \frac{k(z - z_0)^{k-1} g(z) + (z - z_0)^k g'(z)}{(z - z_0)^k g(z)} = \frac{k}{z - z_0} + \frac{g'(z)}{g(z)} \tag{5.11}$$

Je provedena integrace této rovnosti podél křivky C. Geometrický popis je takový, že okolo bodu z_0 je opsána křivka, zvolí se směr a integruje se.

$$\int_C \frac{f'(z)}{f(z)} dz = \int_C \frac{k}{z - z_0} dz + \int_C \frac{g'z}{g(z)} dz \tag{5.12}$$

Funkce $g(z) \neq 0$ na okolí z_0, dále $g(z)$ je holomorfní a totéž platí pro její derivaci. Podíl dvou holomorfních funkcí je opět holomorfní funkce [14]. Podle Cauchyho věty je integrál holomorfní funkce přes uzavřenou jednoduchou křivku C nulový. Pak tedy

$$\int_C \frac{f'(z)}{f(z)} dz = k \int_C \frac{1}{z - z_0} dz + 0 \tag{5.13}$$

Speciálním případem Cauchyho integrálního vzorce je vztah

$$\int_C \frac{1}{z - z_0} dz = 2\pi j \tag{5.14}$$

Tento vzorec lze použít v odvozovaném vzorci:

$$\int_C \frac{1}{z - z_0} dz = 2\pi j = k2\pi j \tag{5.15}$$

$$\frac{1}{2\pi j} \int_C \frac{1}{z - z_0} dz = k \tag{5.16}$$

Porovnáním s větou 2, je výsledkem ten fakt, že číslo k je násobností kořene. Tento postup se tak může aplikovat na libovolný nulový bod (kořen) funkce $f(z)$.

5.2 Věta o poloze kořenů

Díky tomuto poznatku lze určit parametry oblasti v komplexní rovině z, ve které se nachází všechny kořeny daného polynomu.

Pro kořeny rovnice dané tímto vztahem

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 = 0, \tag{5.17}$$

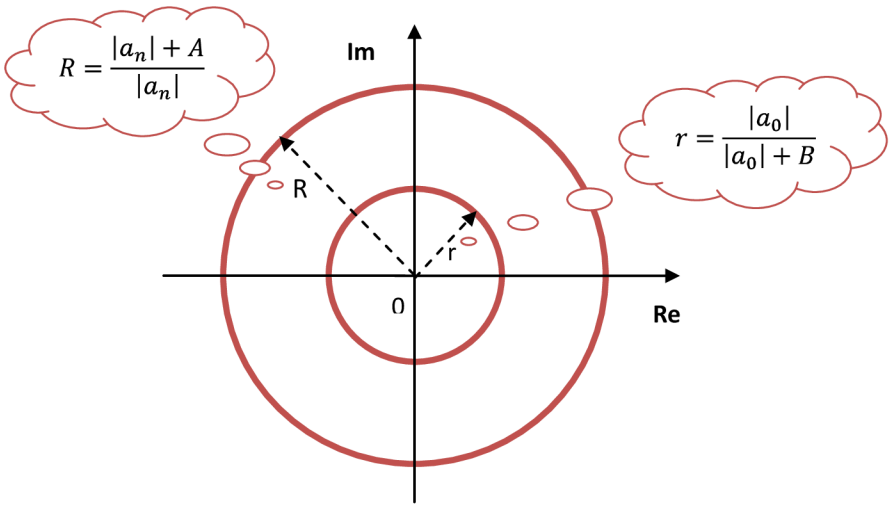
kde $a_n \neq 0$, platí [5], [4]

$$\frac{|a_0|}{|a_0| + B} \leq x_k \leq \frac{|a_n| + A}{|a_n|}, \tag{5.18}$$

kde $k = 1, ..., n$ $A = max(|an - 1|, ..., |a_0|), B = max(|a_n|, ..., |a_1|)$ [5].
Věta o poloze kořenů definuje oblast ohraničenou kružnicí, která je dána podle následujícího vztahu (5.19). Vnitřní oblast u MPA není využita z důvodu zachování jednoduchosti.

$$r = \frac{|a_n| + A}{|a_n|}, \tag{5.19}$$

kde $A = max(|a_{n-1}, \ldots, |a_0)$.



Obrázek 5.2: Věta o poloze kořenů..

Věta o poloze kořenů může být zjednodušena, avšak tento krok není u MPA uplatněn. Důvodem je zvýšení přesnosti výpočtu na úkor časové náročnosti. Jsou známé i další metody určování polohy výskytu kořenů jako například Tillotova nebo Langrangeova věta, dosahujících stejných nebo velice podobných výsledků.

6 SPOJITÉ DYNAMICKÉ SYSTÉMY

Tato kapitola se zabývá vlastnostmi spojitých dynamických systémů (SDS) a také vztahy mezi jednotlivými popisy.

Vnitřní popis spojitých lineárních systémů je popis dynamických systémů pomocí relace vstup-výstup neumožňuje zkoumat děje probíhající uvnitř systému. V některých případech je vnější popis postačující a dokonce výhodnější než vnitřní popis. Platí to zejména o lineárních systémech úlohách, ve kterých není třeba se zabývat energetickou bilancí systému. Práce s operátorovými přenosy zjednodušuje podstatně veškeré výpočty. To ovšem platí za předpokladu nulových počátečních podmínek. Proto byla začátkem šedesátých let vypracována teorie popisu systémů založená na pojmu *stav systému*. Odtud plyne název stavová teorie. Veškeré děje jsou zkoumány v časové oblasti, bez zřetele na frekvenční nebo operátorové souvislosti.

Vnější popis se používá pro vyjádření dynamických vlastností systémů vztahy mezi výstupními a vstupními veličinami. Vztah mezi vstupy a výstupy systému se vyjadřují buď analyticky, pomocí časových odezev na předem definované tvary vstupních signálů nebo frekvenčními vlastnostmi.

Dynamické vlastnosti lze popsat některým z následujících způsobů:

- diferenciální rovnicí,
- přenosovou funkcí (operátorový přenos),
- frekvenčním přenosem,
- frekvenční charakteristikou,
- impulsní charakteristikou (což je časová odezva na Diracův impuls),
- přechodovou charakteristikou,
- rozložením nul a pólů.

6.1 Vztahy mezi popisy systému

Vzájemné vztahy mezi různými vlastnostmi jsou formami vnějšího popisu. Předem nelze jednoznačně říci, která forma popisu je nejvhodnější. Záleží na celé řadě okolností například z jakého zdroje a jakou metodou byly získány informace o systému, co je účelem analýzy či syntézy systému a jaké prostředky jsou k dispozici. Proto je

důležité znát převodní vztahy mezi jednotlivými způsoby popisu a s tím související přechod z jednoho stavu do druhého.

Snadný je přechod od diferenciální rovnice k operátorovému přenosu. Protože přenos je definován za nulových počátečních podmínek, je obraz derivací dán pouze součinem obrazu funkce a příslušných mocnin operátoru p. Stejně jednoduchý je přechod od operátorového přenosu k přenosu frekvenčnímu. Formálně je to provedeno prostou záměnou operátoru p výrazem $j\omega$. Rovněž vykreslení frekvenční charakteristiky ze známého přenosu je rychlé za použití PC.

Složitější je obrácená úloha, tedy k naměřené frekvenční charakteristice stanovit odpovídající přenos. Tato úloha se většinou řeší přibližnou aproximací. Výchozí je obvykle řád aproximačního přenosu, nebo požadovaná přesnost aproximace. Frekvenční charakteristika se nakreslí v logaritmických souřadnicích a hledá se takový řád a koeficienty přenosové funkce, které zajistí potřebnou shodu jak ampitudy, tak fáze v uvažovaném frekvenčním rozsahu. Celý postup je jen těžko algoritmizovatelný a řešitelný pomocí výpočetní techniky [7].

Vzájemné vztahy mezi časovými charakteristikami, impulsní a přechodovou, jsou rovněž jednoduché. Pro impulsní odezvu platí $g(t) = L^{-1}\{F(p)\}$, kdežto pro přechodovou charakteristiku $h(t)$ platí $h(t) = L^{-1}\{F(p)\frac{1}{p}\}$. Odtud je zřejmé, že impulsní charakteristika je derivací přechodové charakteristiky $g(t) = \frac{dh(t)}{dt}$ a obráceně přechodová charakteristika je integrální funkcí impulsní charakteristiky $h(t) = \int_0^t g(t)dt$ [7].

6.2 Lineární diferencilální rovnice

Lieární, stacionární a spojitý systém se vstupem $u(t)$ a výstupem $y(t)$ popisuje lineární diferenciální rovnice s konstantními koeficienty [7].

$$a_n y^{(n)}(t)+a_{n-1}y^{(n-1)}(t)+...+a_1 y(t)+a_0 y(t) = b_m u^{(m)}(t)+...+b_1 u(t)+b_0 u(t) \quad (6.1)$$

kde a_i, b_i jsou reálné konstanty, $u(t)$ vstup systému a $y(t)$ je výstup systému [7]. Rovnici (2.1) lze zapsat ve tvaru:

$$\sum_{i=0}^{n} a_i y^{(i)}(t) = \sum_{j=0}^{m} b_j u^{(j)}(t) \quad (6.2)$$

6.3 Operátorový přenos

Definice operátorového přenosu. Operátorový přenos je dán poměrem obrazu výstupní veličiny k obrazu vstupní veličiny ve stejné transformaci, za předpokladu nulových

počátečních podmínek. V případě spojitých systému je používána Laplaceova transformace. Systém popsaný diferenciální rovnicí, má přenos ve tvaru racionální lomenné funkce

$$F(p) = \frac{Y(p)}{U(p)} = \frac{b_m p^m + b_{m-1} p^{m-1} + ... + b_1 p + b_0}{a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p + a_0} \tag{6.3}$$

kde p je Laplaceův operátor. Z uvedené podmínky realizovatelnosti plyne, že stupeň polynomu v čitateli musí být nižší, nebo nejvýše roven stupni polynomu ve jmenovateli přenosu $F(p)$.

Oba polynomy lze vyjádřit ve tvaru součinu kořenových činitelů

$$A(p) = a_n(p - p_1)(p - p_2)...(p - p_n) \tag{6.4}$$

$$B(p) = b_m(p - n_1)(p - n_2)...(p - n_n) \tag{6.5}$$

Obecně se komplexní čísla p_i, $i = 1..n$ nazývají póly přenosu, nebot splňují rovnici $A(p_i) = 0$.

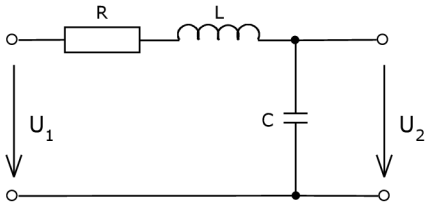
Rovněž obecně komplexní čísla n_j, $j = 1..m$ jsou nuly přenosu, pro které platí $B(n_j) = 0$.

Pomocí těchto tvarů lze přenos psát ve formě podílů kořenových činitelů

$$F(p) = \frac{b_m(p - n_1)(p - n_2)...(p - n_m)}{a_n(p - p_1)(p - p_2)...(p - p_n)} \tag{6.6}$$

kde p je Laplaceův operátor [7].

Příklad jednoduchého RLC obvodu popsaného operátorovým přenosem



Obrázek 6.1: Jednoduchý RLC obvod

Přenos článku na obrázku 6.1 je dán vztahem

$$U_2 = U_1 \frac{Z_2}{Z_1 + Z_2} \tag{6.7}$$

kde U_2 je výstupní napětí, U_1 je vstupní napětí a Z_2 je impedance obvodu z pohledu výstupních svorek. U tohoto příkladu je tedy $Z_2 = \frac{1}{j\omega C}$ a $Z_1 = R + j\omega L$. Zavedeme

substituci $s = j\omega$ a přenos se vypočítá pomocí vztahu $K(s) = \frac{U_2}{U_1}$. Po dosazení do vztahu pro výpočet výstupního napětí děliče vyjde vztah pro přenos:

$$K(s) = \frac{U_2}{U_1} = \frac{Z_2}{Z_1 + Z_2} = \frac{\frac{1}{sC}}{R + sL + \frac{1}{sC}} \tag{6.8}$$

$$K(s) = \frac{1}{s^2LC + sCR + 1} \tag{6.9}$$

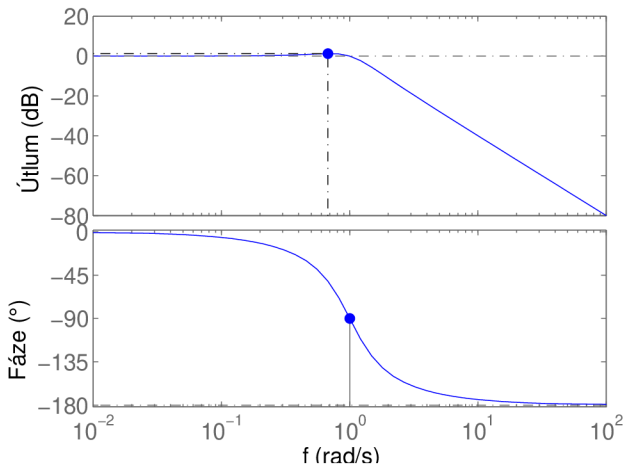
pokud hodnoty všech součástek budou rovny jedné $\Longrightarrow$ výsledný vztah bude mít tvar:

$$K(s) = \frac{1}{s^2 + s + 1} \tag{6.10}$$

Z tohoto vztahu jsou přímo vidět nuly a póly daného obvodu (nuly jsou kořeny čitatele a póly jsou kořeny jmenovatele) $p_1 = p_2^*$

$$\begin{aligned} p_1 &= -0.5 + j0.866 \\ p_2 &= -0.5 - j0.866 \end{aligned} \tag{6.11}$$

Tyto výsledky jsou doloženy z programu MATLAB a ověřeny obvodovým simulátorem PSPICE.



Obrázek 6.2: Frekvenční charakteristiky RLC obvodu.

6.4 Frekvenční přenos systému

Vyjadřuje vlastnosti systému pro harmonický proměnný vstupní signál. Frekvenční přenos je roven podílu Fourierova obrazu výstupního signálu a Fourierova obrazu

vstupního signálu. Frekvenční přenos pak udává amplitudové zesílení a fázové natočení procházejícího signálu [7]:

$$F(j\omega) = \frac{Y(j\omega)}{U(j\omega)} = |F(j\omega)| \cdot e^{j\varphi(\omega)} \tag{6.12}$$

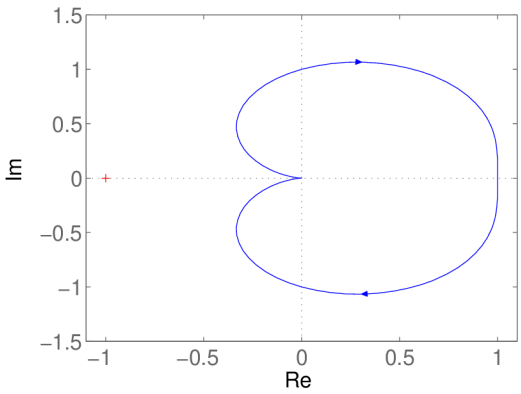
6.5 Frekvenční charakteristika

Je grafické vyjádření frekvenčního přenosu systému, lze ji nakreslit ze známého frekvenčního přenosu, nebo pomocí hodnot, změřených na skutečném systému. Vektor frekvenčního přenosu může být vyjádřen dvěma způsoby [7]:

$$F(j\omega) = Re[F(j\omega)] + jIm[F(j\omega)] \tag{6.13}$$

V tomto případě je obvyklé kreslit frekvenční charakteristiku v komplexní rovině s osami, na které se vynáší reálná a imaginární část přenosu. Frekvenční vlastosti sytému vyjadřuje křivka v komplexní rovině, jejímž parametrem je kruhová frekvence označená jako ω.

$$F(j\omega) = |F(j\omega)|e^{j\varphi(\omega)} \tag{6.14}$$



Obrázek 6.3: Frekvenční charakteristika v komplexní rovině tzv. Nyquistův diagram.

Vlastnosti systému nyní určují dvě funkce a jim odpovídající dvě křivky. První z nich je závislost absolutní hodnoty přenosu na frekvenci a druhá vyjadřuje průběh fáze. Pro práci s amplitudovými charakteristikami je vhodné volit logaritmické měřítko, amplituda je pak vyjadřena v decibelech [7]:

$$|F(j\omega)|_{dB} = 20log|F(j\omega)|. \tag{6.15}$$

Frekvenční charakteristiku lze nakreslit ze známého frekvenčního přenosu, nebo pomocí hodnot, změřených na skutečném systému. Tento způsob se u systémů s malými časovými konstantami ještě stále používá (zejména ve sdělovací a telekomunikační technice).

6.6 Impulsní charakteristika

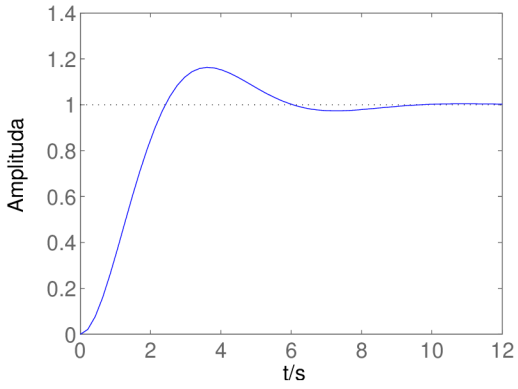
Je odezva dynamického systému na vstupní signál tvaru Diracova impulsu při nulových počátečních podmínkách. Impulsní charakteristiku systému lze použít i k výpočtu odezvy systému na libovolný vstupní signál. Funkce se označuje jako $g(t)$. Diracův impuls $\delta(t)$ je nerealizovatelná funkce, definovaná následujícími vztahy [7]:

$$\delta(t) = \begin{cases} 0 & \text{pro } t \neq 0 \\ 1 & \text{pro } t = 0 \end{cases} \tag{6.16}$$

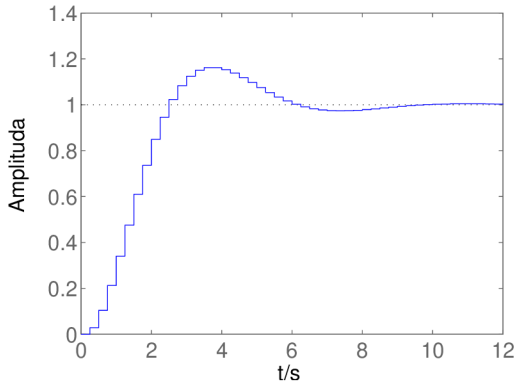
$$\int_{-\infty}^{+\infty} \delta(t)dt = 1 \tag{6.17}$$

6.7 Přechodová charakteristika

Přechodová charakteristika je další časová funkce, kterou se často vyjadřují dynamické vlastnosti systému. Je to odezva na jednotkovou změnu (jednotkový skok) vstupní veličiny při nulových počátečních podmínkách. Obvykle se značí $h(t)$ [7].



Obrázek 6.4: Přechod. charakteristika.



Obrázek 6.5: Diskrétní popis.

Jednotkový skok je definován takto:

$$u_0(t) = \begin{cases} 0 & \text{pro } t < 0 \\ 1 & \text{pro } t \geq 0 \end{cases} \tag{6.18}$$

Laplaceův obraz této funkce je

$$U_0(p) = L\{u_0(t)\} = \frac{1}{p} \tag{6.19}$$

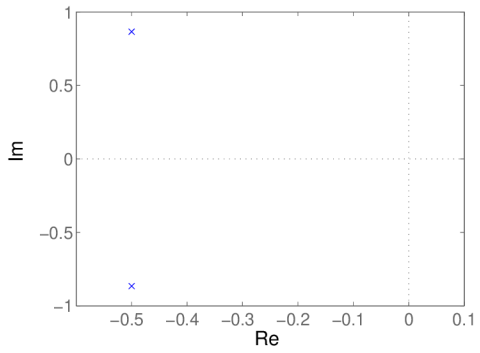
a pro obraz přechodové charakteristiky tedy platí

$$H(p) = L\{h(t)\} = \frac{1}{p}F(p) \tag{6.20}$$

Pomocí přechodové charakteristiky lze vypočítat odezvu na obecný vstupní signál $u(t)$.

6.8 Rozložení nul a pólů přenosu

U pólů a nul je rozhodující jejich poloha vzhledem k imaginární ose (mez stability). V levé komplexní polorovině jsou stabilní póly a nuly (mají zápornou reálnou část), v pravé polorovině jsou nestabilní póly a nuly (mají kladnou reálnou část). Stabilní nula způsobuje **překmit** a nestabilní nula způsobuje **podkmit** děje. [7], [3],[8].



Obrázek 6.6: Rozložení nul a pólů RLC obvodu.

Nuly v počátku představují derivační charakter systému, póly v počátku naopak představují integrační charakter. Příčinou záporných reálných pólů je aperiodický přechodový jev. Komplexně sdružené póly způsobují kmitavý charakter přechodového děje. Čím jsou stabilní póly dále od imaginární osy, tím je přechodový děj více tlumen. Jsou-li nuly blíže imaginární ose než póly, bude převládat derivační charakter. Protože stabilní oblastí u diskrétních systémů je prostor uvnitř jednotkové kružnice, je zde rozhodující sledovat polohu pólů a nul právě vůči této kružnici (mez stability) [15].

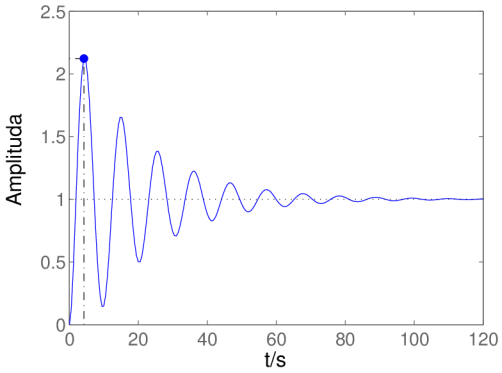
6.9 Stabilita systému

Stabilita je jedním ze základních požadavků, který je kladen na regulační obvod. Regulační obvod je stabilní, jestliže pro vychýlení regulačního obvodu z rovnovážného stavu a odeznění vnějších sil, které tuto odchylku způsobily, se regulační obvod během času znovu vrátí do původního rovnovážného stavu. Matematicky lze stabilitu definovat [7]:

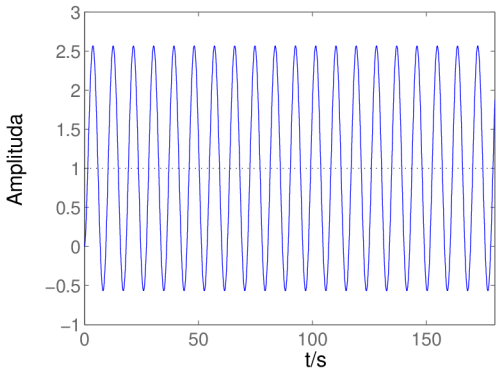
$$\lim_{t\to\infty} y(t) = 0. \tag{6.21}$$

Z hlediska stability existují regulační obvody stabilní, na mezi stability a nestabilní. Regulační obvody na mezi stability se považují za stabilní. Vždy se vyžaduje, aby regulační obvod byl za všech okolností stabilní. Zatímco parametry a dynamické

vlastnosti regulované soustavy jsou dány konstrukcí soustavy, technologickým procesem apod. (nemohou se měnit), dynamické vlastnosti regulátoru nastavováním volitelných parametrů regulátoru se změnit mohou. Tím je dosaženo stability (a dalších vlastností) regulačního obvodu.

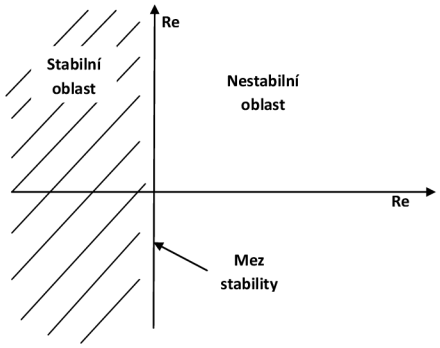


Obrázek 6.7: Přechodová charakteristika stabilního systému.



Obrázek 6.8: Přechodová charakteristika systému na mezi stability.

Kontrola stability RO spočívá v určení rozložení kořenů charakteristické rovnice v komplexní rovině kořenů. Pokud lze kořeny vyčíslit, použije se nutná a postačující podmínka stability, která se týká kořenů charakteristické rovnice. *Regulační obvod je stabilní právě tehdy, když všechny kořeny charakteristické rovnice mají záporné reálné části, tedy leží-li v levé komplexní polorovině.*



Obrázek 6.9: Postačující podmínka stability.

Jinak je nutno použít pravidla, která umožní rozhodnout o stabilitě bez přímého výpočtu kořenů, tyto pravidla se nazývají kritéria stability. Mezi často používaná kritéria patří například Nyquistovo, Michajlovovo nebo Hurwitzovo kritérium.

7 ALGORITMY VÝPOČTU

Tato kapitola popisuje kompletní postup programování a ladění funkce pro výpočet kořenů polynomů metodou přírustku argumentu. Algoritmus lze rozdělit do tří částí.

První část se zabývá určením vstupních parametrů funkce ze zadaného polynomu a výpočtem celkového počtu kořenů. To znamená například zjištění řádu polynomu nebo hodnoty maximálního koeficientu pro určení potřebného poloměru (5.19). Druhá část řeší lokalizaci všech kořenů zadaného polynomu Newtonovou metodou tečen a navazuje na první část. Třetí a poslední blok tvoří úsek programu, který obstarává určení násobnosti jednotlivých nalezených kořenů.

Podmínkou správnosti výpočtu je, aby součet násobností všech kořenů byl roven celkovému počtu kořenů. Pokud tato podmínka není splněna nastala tak ve výpočtu chyba.

7.1 Funkce určující celkový počet kořenů

Naprogramovaná funkce vrací počet všech kořenů, které se nacházejí uvnitř transformované kružnice, tedy celkový počet kořenů polynomu. Odpověď na otázku proč byla křivkou vybrána právě kružnice dáva vztah (5.19). Kružnice je dána dvěma vstupními parametry: polynomem a přesností výpočtu označovaná jako ϵ. Tyto dva jedinné důležité údaje zadává uživatel z klávesnice.

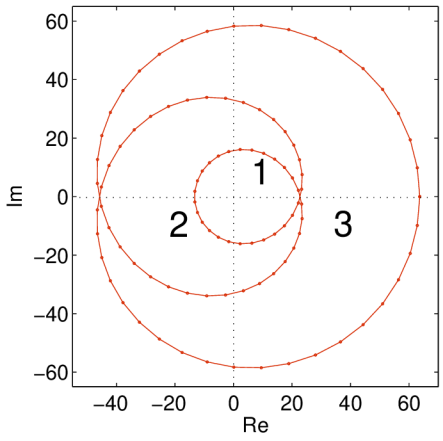
Celý postup se odvíjí od zadaných vstupních parametrů pevně danými kroky. Transformovaná kružnice je rozdělena například na n ekvidistantně vzdálených části a vznikne tímto způsobem vektor n úhlů. Kolik takových úseků vznikne záleží na řádu daného polynomu.

```
temp = linspace(0,2*pi,rozliseni);
bodykruznice = polomer*exp(temp*sqrt(-1));
```

Pro každý takovýto bod na kružnici daný úhlem a souřadnicemi v komplexní rovině z, se vypočítá hodnota polynomu v tomto bodě. Výsledkem operace je další vektor dat, který už náleží do roviny w, nikoliv do roviny z a proběhla tedy transformace z komplexní roviny z do komplexní roviny w.

```
w = polyval(polynom,bodykruznice);
```

Vykreslením takového vektoru do komplexní roviny vznikne zajímavý pohled, který je vidět na obrázku 7.1. Při bližším nastudování princincipu přírustku argumentu a daného obrázku je dobře patrné, kolik kořenů má zadaný polynom. Celkový počet kořenů je roven počtu otáček kolem počátku soustavy souřadnic komplexní roviny w.



Obrázek 7.1: Zjištění násobnosti kořene.

Na první pohled velice jednoduchá situace, avšak z hlediska programování nikoliv. Aby bylo možné zjistit počet oběhů kolem počátku soustavy souřadnic, byl navržen nový postup výpočtu.

1. Po transformaci a vykreslení v rovině w je vypočítána fáze jednotlivých bodů v této rovině.
2. Dalším krokem je provedení korekce úhlů z intervalu $\langle -\pi, +\pi \rangle$ do intervalu $\langle 0, 2\pi \rangle$ (volitelně je možné pracovat i ve stupňové míře).
3. Poté je provedena diference dvou sousedních prvků v matici úhlů.
4. Tím je vytvořena další matice prvků tentokráte obsahující jenom diference sousedních prvků.
5. Z takového pole jsou odstraněny fiktivní oběhy, tj. za oběh jsou brány diference splňující podmínku: $diference > 270°$.
6. Předposledním krokem je zjištění zda je rozdíl diferencí kladný nebo záporný (za kořeny jsou brány $diference < 0$), tím vzniká opět nová matice prvků.
7. Nyní stačí zjistit počet záporných prvků v předchozí uvedené matici.
8. Výsledkem je tedy celkový počet kořenů polynomu, v tuto chvíli nejsou známy jejich číselné hodnoty.

Těmito uvedenými kroky končí první části výpočtu kořenů metodou přírustku argumentu, je tedy v této fázi vyřešen celkový počet kořenů daného polynomu.

7.2 Funkce pro nalezení kořenů

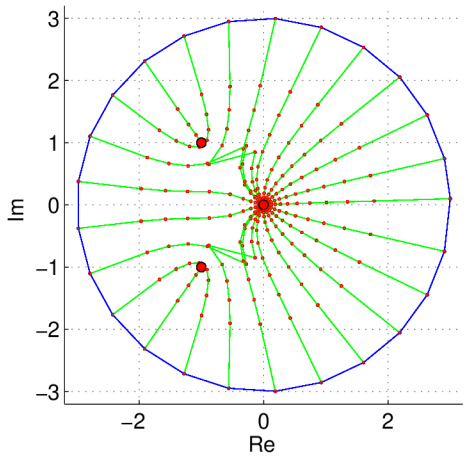
Tato funkce by měla rychle a hlavně přesně najít všechny kořeny polynomu. Pro lokalizaci kořene byla vybrána Newton-Raphsonova metoda tečen. Postup při programování vychází z teoretických poznatků o jednobodové Newtonově metodě tečen. Vývojový diagram této metody je uveden v příloze *D.1* této práce. Úvodní kroky jsou situovány takto:

1. Stanovení potřebného rozlišení, tj. na kolik ekvidistantně vzdálených bodů má být kružnice rozdělena.

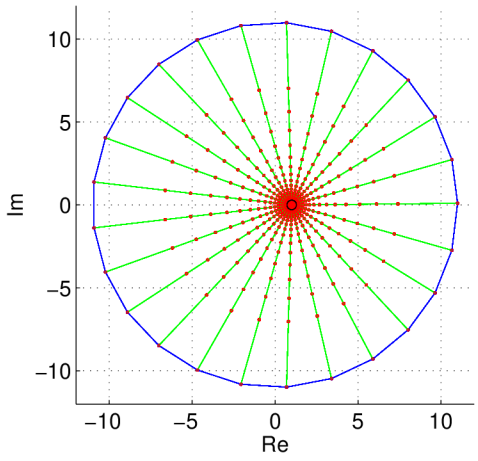
2. Dle věty o poloze kořenů je stanoven potřebný poloměr kružnice, dáno vztahem (5.19). Aby nebyla náročnost výpočtu vysoká je při překročení maximálního poloměru automaticky přidělen poloměr hraniční, jde tedy pomocnou konstantu.

3. Velice důležitým krokem je přidělení imaginární jednotky o malé velikosti prvnímu a poslednímu bodu kružnice, protože jsou to pouze reálná čísla. Přiřazení je dáno prostým důvodem: z reálného počátečního bodu může být nalezen zase jen kořen reálný. V případě, že by polynom měl pouze imaginární kořeny tak výpočet zkolabuje. Příkladem je polynom $x^2 + 2x + 1 = 0$.

4. Následuje poslední část, a to samotná lokalizace kořenů Newton-Raphsonovou metodou tečen, která si zaslouží podrobnější popis.



Obrázek 7.2: Lokalizace kořenů polynomu $x^5 + 2x^4 + 2x^3 = 0$ v komplexní rovině z.



Obrázek 7.3: Lokalizace kořenů polynomu $(x - 1)^5 = 0$ v komplexní rovině z.

Celý výpočet probíha v těle *for* cyklu. Ten proběhne tolikrát, na kolik částí je rozdělena kružnice. Každý bod na kružnici je počátečním bodem aproximace pro Newton-Raphsonovu metodu tečen.

```
for pocitadlo=1:length(bodykruznice)
    pocetiteraci=0;
    korenyprubezne=0;
    x0=bodykruznice(pocitadlo);
    ...
```

Postup pokračuje derivací funkce (polynomu) a poté je pomocí aproximačního vztahu (4.1) vyjádřen první krok iterace. Pokud byl počáteční bod zvolen nejlépe jak mohl, může být kořen polynomu nalezen po prvním iteračním kroku. Pokud se tak nestalo, probíhá v cyklu *while* výše uvedený výpočet do té doby, než je splněna podmínka $|x_1 - x_0| > \epsilon$. Celý zmíněný postup lokalizace je znázorněn na vývojovém diagramu v příloze *D.1*. Ukázková část zdrojového kódu pomocí níž než kořen nadále zpřesňován, je vypsána zde:

```
while abs(x1-x0) > presnost
      x0=x1;
      pocetiteraci=pocetiteraci+1;
      x1=x0-(polyval(polynom,x0)/polyval(prvniderivace,x0));
      korenyprubezne(pocetiteraci)=x1;
end
```

Pojem přesnost není možné zaměňovat s pojmem chyba řešení, která je dána rozdílem mezi hodnotou vypočítanou a skutečnou a charakterizuje správnost řešení. Protože skutečná hodnota výsledku není známa, nelze o chybě řešení vůbec hovořit. Může nastat situace, kdy výsledek řešení bude přesný, ale zdaleka nebude správný. Při praktickém zadávání hodnoty požadované přesnosti řešení se musí vždy brát v úvahu prostředek na kterém je prováděn výpočet. Každý typ počítače je schopen zobrazit definovanou malou hodnotu. Každá další menší hodnota je pak již interpretována jako nula. Takže se může stát, že požadovaná přesnost bude menší než je tato limitní hodnota, potom se ve výpočetním programu projeví jako nula. V iteračním výpočtu to povede k nekonečnému počtu opakování, program se tzv. zacyklí a prakticky nemá možnost ukončení.

Všechny nalezené kořeny (jednotlivé poslední iterace) jsou uloženy do nově vytvořené matice, s kterou se nadále pokračuje ve výpočtech.

7.3 Ověření násobnosti kořene

Upřesnění kořenů má opodstatnění hlavně při hledání násobných kořenů. Proč tomu tak je, bude vysvětleno dále. Protože aplikací Newton-Raphsonovy metody tečen

byl u násobných kořenů nalezen shluku blízkých kořenů, je nutné zajistit určení násobnosti jednoho jedninného kořene. Matematicky lze postup provést jako vypočet geometrického středu shluku kořenů a ověřit MPA jeho násobnost.

Protože jednotlivými iteracemi jsou nalezeny různé kořeny polynomu, je výsledkem hledání pole neseřazených kořenů (matice z předchozí kapitoly). Proto je žádoucí toto pole seřadit z důvodu určení počtu nalezení jednotlivých kořenů.

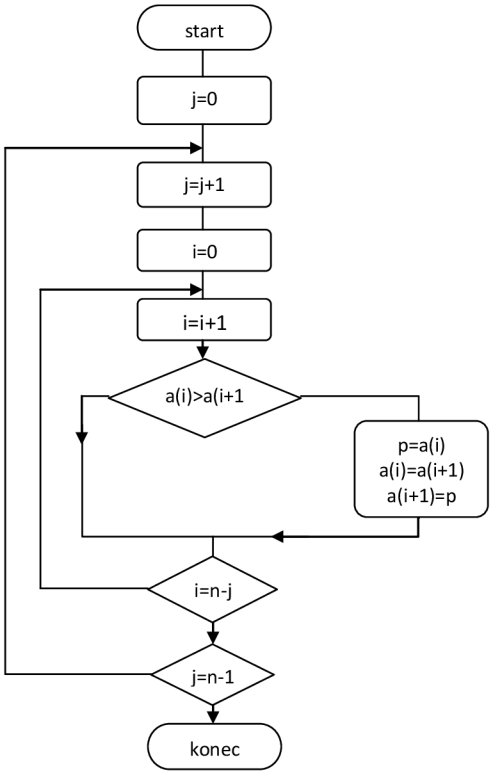
Prvním způsobem jak seřadit matici reálných, komplexních nebo smíšených kořenů je pomocí příkazu programu Matlabu. Jedná se o příkaz *sort*. Protože tento příkaz dokáže řadit pouze a jen podle jednoho kritéria (buď podle reálných nebo imaginárních částí), je dále využita na takto vytvořenou z části seřazenou matici metoda Buble sort.

Princip spočívá v tom, že jsou porovnávany dva sousední prvky podle velikostí. Pokud je první hodnota větší než druhá, tak se tyto dva prvky prohodí. Aplikací postupu na celé pole hodnot se v posledním kroku největší prvek dostává na konec matice. Poté se pole o tento prvek zkrátí, protože nejvetší hodnota se dostala nakonec a není tudíž zapotřebí. Celý postup se opakuje $n-1$, kde n udává počet prvků v poli.

Prvním způsobem je tedy seřazeno pole podle reálných částí a po použití druhé metody je pole seřazeno navíc ještě podle imaginárních částí. Tímto algoritmem je docíleno seřazení stejných kořenů v matici vedle sebe.

Vystáva zde problém, kde se nachází další rozdílný kořen. Problém je vyřešen zjištěním indexu na prvek matice, který je odlišný od prvku předchozího. Díky těmto indexům je snadné separovat potřebné kořeny z matice. V uvedeném zdrojovém kódu reprezentuje indexy proměnná s názvem *delka—hledej*. Následně jsou vypočítány sumy stejných kořenů, které odpovídají proměnné *geom—pole*.

```
for temp2=1:delka_hledej
    for ukazatel=temp:pole_hledej(k)
        prubezny_vysledek=real(B(ukazatel))+sqrt(-1)*imag(B(ukazatel));
        vysledek=prubezny_vysledek+vysledek;
        pocitadlo=pocitadlo+1;
    end
        pocitac(temp2)=pocitadlo;
        pocitadlo=0;
        geom_pole(temp2)=vysledek;
        k=k+1;
        n=n+1;
        temp=pole_hledej(n)+1;
        vysledek=0;
        prubezny_vysledek=0;
end
```



Obrázek 7.4: Vývojový diagram metody Buble sort.

Po těchto operacích už následuje výpočet geometrických středů. Aby byla hodnota geometrického středu správná, musí se vypočítat jako součet reálných částí a imaginárních částí dělený počtem kořenů ve shluku.

```
for temp=1:delka_hledej
    geometricke_stredy(temp)=geom_pole(temp)/pocitac(temp);
end
```

V takto vytvořených geometrických středech je opět aplikována metoda přírustku argumentu, která zjistí násobnost kořene. Je vidět, že metoda přírustku argumentu je využita ve dvou fázích výpočtů:

- Prvním využitím je zjištění celkového počtu kořenů nacházejících se uvnitř transformované kružnice.
- Druhou aplikací je zjištění (ověření) násobnosti kořene.

Vypočtené výsledky jsou následně využity uživatelským prostředím, které je prezentuje ve vizuální podobě.

8 TVORBA GRAFICKÉHO ROZHRANÍ

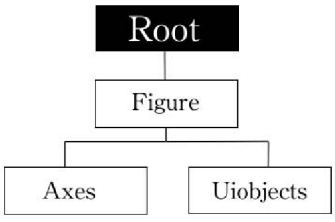
Všechny grafické objekty dané aplikace tvoří grafické uživatelské prostředí. Název je převzat z anglického Graphical User Interface (GUI). Většina lidí v dnešní době dává přednost aplikaci ve které stačí k používání pouze myš, stačí kliknout myší a hned tato akce vyvolá nějakou událost. Práce je řízena interaktivními kroky.

Tato kapitola by měla zavést do možností grafického prostředí Matlabu. Všechny grafické objekty jsou vytvářeny přímo v editoru zdrojového kódu. V takovém případě lze psát, že GUI je optimální, protože takové prostředí obsahuje z hlediska programátora jen nejnutnější komponenty pro spávnou funkčnost. Programovém prostředí Matlab umožňuje výběr ze dvou možností jakým způsobem gafickou aplikaci naprogramovat.

Prvním způsobem je použití nástroje, který vytvoří všechny grafické prvky pouhým přetažením, jedná se o nízkoúrovňové programování, protože po sestavení aplikace je automaticky vygenerován příslušný zdrojový kód. Tento nástroj se jmenuje GUIDE (Graphical User Interface Development Environment).

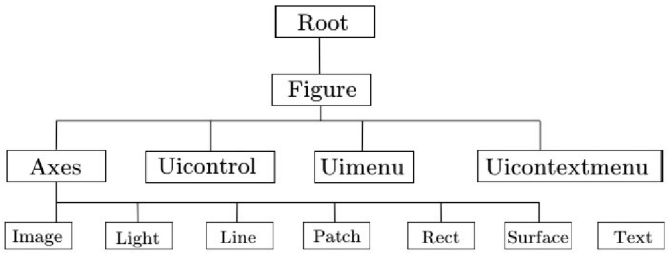
Postup vytváření je dán pevnými, intuitivními kroky. Tedy použití tohoto způsobu je univerzální a časově nenáročné. Nevýhodou tohoto způsobu je možná nekompatibilita se staršími verzemi programového prostředí Matlab a zdrojový kód nemusí a často není optimální.

Druhou možností řešení je využití znalostí systému Handle Graphics. Nástroj s jehož pomocí lze efektivně pracovat s grafickými objekty. Obrovskou výhodou tohoto způsobu je absolutní kontrola programátora nad aplikácí narozdíl od předchozího způsobu programování. Grafickými objekty jsou všechny základní elementy grafického výstupu. Aby systém fungoval správně musí být mezi jednotlivými objekty přesně daná hiearchie, která je zobrazena na následujícím obrázku.



Obrázek 8.1: Základní hierchie Matlabu.

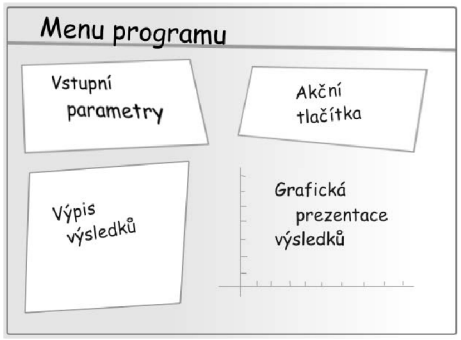
S pojmem hiearchie objektů souvisí pojmy Parent (rodič) a Children (děti). Pokud je provedena změna na vyšší úrovni, tak tato změna může být zděděna nižšími podřízenými objekty viz. hiearchie grafických objektů [12].



Obrázek 8.2: Hiearchie základních grafických objektů v Matlabu.

8.1 Návrh aplikace

Před vlastním programováním daného grafického prostředí je vždy vhodné rozhodnout se jak by měla daná aplikace vypadat. Z tohoto důvodu je užitečné nakreslit si návrh na papír a ten poté realizovat v prostředí Matlab. Jedná se o možná nejefektivnější vývojový proces před vlastní prací v programovém prostředí. Protože při programování je uplatněna metoda programování s názvem *Switch Board* je základem podstatné využití vlastnosti funkcí, a to možnost volat sama sebe s různými parametry a v neposlední řadě využití příkazů Switch a Case. Předpokladem je znalost systému Handle Graphics.

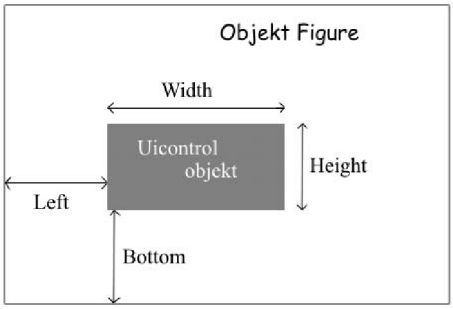


Obrázek 8.3: Návrh grafického prostředí.

Rozložení aplikace je navrženo tak, aby bylo co nejlépe přehledné a nečinilo tak uživateli problémy s ovládáním. Předběžná vizuální podoba programu je na obrázku 8.3. Z funkčního hlediska by navrhovaná aplikace měla mít několik částí navzájem od sebe oddělených. Oddělení je důležité pro odlišení prvků s různou funkčností. Tímto pojmem je rozuměno například oddělení uicontrol objektů, které slouží jako ovládání nebo mají pouze informační charakter (například výsledková část).

Jelikož byla vybrána za metodu programování technika *Switch Board*, musí jednotlivé objekty úmístěny na správné místo. K tomu slouží pozicování, kdy každý

prvek jem možné nadefinovat v aktuálním okně Figure pomocí čtyř parametrů, jak ukazuje obrázek.



Obrázek 8.4: Pozicování objektů.

left – umístění levého spodního rohu grafického objektu od počátku

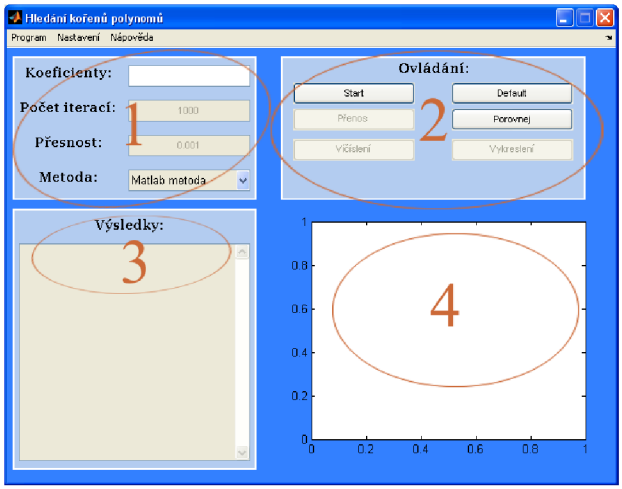
right – umístění pravého spodního rohu grafického objektu od počátku

width – šířka uicontrol objektu

height – výška uicontrol objektu

8.2 Popis částí aplikace

V nově vytvořeném okně Figure se nachází všechny použité uicontrol prvky. Souhrn několika takovýchto prvků tvoří celkem čtyři základní bloky programu, mezi něž není zařazeno *menu* programu. Rozpoložení je na obrázku 8.5.



Obrázek 8.5: Navržené uživatelské prostředí.

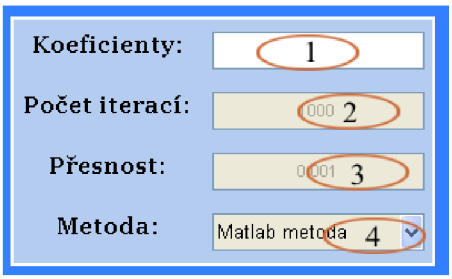
Popis první části GUI

Blok programu umožňuje zadávání všech vstupních parametrů nutných pro správný výpočet. Jedná se o tři editovací pole a rozbalovací *popup-menu*. Prvním editovacím polem je *zadávání koeficientů* polynomů, kde funkce odpovídá názvu. U zatržené volby v menu programu *výpočet přenosové funkce* plní toto pole stejnou úlohu.

Druhé editovací pole slouží k zadávání maximálního počtu iterací. Hodnota je vstupním parametrem použitých metod (omezeno na některé metody). V případě výpočtu přenosové funkce plní stejnou úlohu jako první editovací pole. Pokud je vybráno v menu programu *vyčíslování* polynomu viz. příloha *C*, slouží toto pole k zadávání hodnoty v níž má být polynom vyčíslen.

Poslední editovací pole umožňuje zadávání přesnosti výpočtu pro použitou metodu přírustku argumentu (jeho hodnota je vstupním parametrem pro naprogramované funkce).

Položky v rolovací nabídce slouží k výběru metody, pomocí které mají být nalezeny všechny kořeny polynomu.



Obrázek 8.6: Blok zadávání vstupních parametrů.

Příklady zadávání vstupních hodnot

1. Pro polynom, který je zadán například takto: $(x-1)^5$ musí být koeficienty zadány v tomto tvaru 1 -5 10 -10 5 -1. Pokud jde o výpočet konkrétního polynomu, toto pole obsahuje kořeny polynomu jež má být vypočítán. Například pokud je zadáno 1 1 1 1 1, je výsledkem polynom s právě těmito kořeny, tedy: $(x-1)^5$.

2. Počet iterací se zadává jako celé číslo, výchozí hodnotou je 1000 (čím větší hodnoty, tím déle trvá výpočet). V případě vykreslení polynomu se do tohoto pole zadává rozsah osy x v němž má být polynom vykreslen. Například rozsah osy může být zadán číselně: $[-3\ 3]$. Pro případ výpočtu konkrétního polynomu je v tomto poli uveden konečný výsledek výpočtu (koeficienty polynomu). U volby *vykreslování*, toto pole obsahuje číselnou hodnotu, v níž má být polynom vyčíslen.

3. Toto editovací pole je využito pro zadání přesnosti výpočtu. Ta se zadává jako desetinné číslo, výchozí hodnotou je 0,001 (platnost nalezení kořene na 3 desetinná místa). Program toleruje zadávání desetinného čísla s tečkou, nebo čárkou. V případě *vykreslování* polynomu slouží toto pole také pro zadání kroku s kterám má být rozdělen daný x-ový rozsah objetku *axes*.

4. Popup-menu má celkem sedm možností jakou metodou mohou být kořeny polynomu vypočítány. Jedná se o tyto metody:

   Matlab metoda – metoda aplikovaná v prostředí Matlab,

   MPA – nově vytvořená metoda na hledání kořenů polynomů,

   Lagurrova metoda – metoda plikovaná ve starších verzích Matlabu,

   Bairstowova metoda – aplikována pro názorné porovnání výsledků

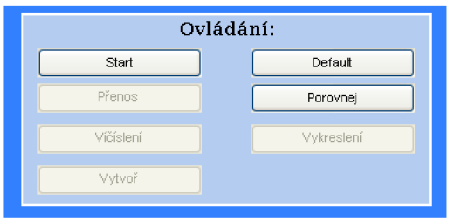
   Halleyova metoda – aplikována pro názorné porovnání výsledků

   Newtonova metoda – klasická metoda bez úprav

   Durand-Kernerova metoda – aplikována pro názorné porovnání výsledků

Popis druhé části GUI

Prováděcí část obsahuje řadu tlačítek. Tato tlačítka fungují jako odezvy na prováděné operace. Po jejich stisknutí je volána příslušná funkce, a ta již vykoná svůj obsah operačního kódu. Tlačítko se po stisknutí vrací zpět do původního stavu, je tedy připraveno k opětovné reakci uživatele. Tento blok obsahuje celkem sedm těchto uicontrol objektů. Příslušné tlačítko je zpřístupněno v závislosti na druhu výpočtu.



Obrázek 8.7: Blok prováděcí části.

Start – začátek výpočtu kořenů polynomu.

Přenos – výpočet nul a pólů přenosové funkce.

Vyčíslení – vyčíslení polynomu v daném bodě.

Default – nastavení programu do počátečního stavu včetně vymazání všech hodnot a průběhů.

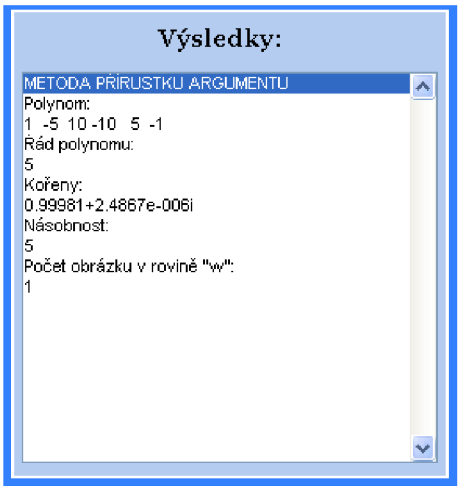
Porovnej – otevření nového okna a porovnání výsledků jednotlivých metod uvedených v Popupmenu.

Vykreslení – vykreslení polynomu na daném intervalu.

Vytvořit – vytvoření polynomu s danými kořeny.

Popis třetí části GUI

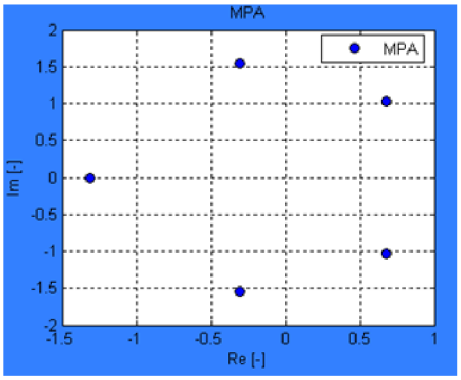
Část grafického prostředí, do kterého se vypisují vypočtené výsledky obsahuje uicontrol objekt *ListBox*. Hlavní využitou vlastností je *String*. Díky ní mohou být všechna data v matici po převodu do spávného formátu zobrazena do *Listboxu*.



Obrázek 8.8: Výpis vypočítaných výsledků do uicontrol objektu.

Popis čtvrté části GUI

Pro grafickou reprezentaci vypočtených výsledků byl vytvořen kreslící prostor tvořený grafickým objektem *Axes*. Tento objekt tvoří poslední ze čtyř hlavních částí programu, vyjíma uživatelského menu.



Obrázek 8.9: Vykreslování do grafického objektu *axes*.

8.3 Programování prostředí

Rozmístěním grafických objektů začíná nastavování jednotlivých vlastností objektů. Mezi takové důležité vlastnosti patří:

```
h6=uicontrol('Units','normalized','Style','Text','String',...
             'Přesnost:','Position',[0.02 0.75 0.16 0.05],...
             'Tag','presnoststatic','Fontsize',12,...
             'FontWeight','bold','BackgroundColor',...
             [0.7 0.8 0.94],'Fontname','Georgia');
```

- Units - nastavení jednotek
- Style - typ uicontrol objektu
- Text - vlastní text, popis objektu
- String - řetězec znaků například v editovacím poli
- Tag - identifikace objektu

Postup vytváření grafického prostředí výchází principielně z tohoto ukázkového bloku programu.

```
switch(vstpar)
      case('vybermetody')
            Volba = get(findobj('Tag','vybermetody'),'Value');
            if Volba == 1
            set(findobj('Tag','presnost'),'Enable','off');
                  ...
            elseif Volba == 2
                  ...
            end
       case('start')
            ...
      end
end
```

Jak je vidět z části zobrazeného zdrojového kódu, příkaz *switch* představuje jakýsi přepínač reagující na odezvu programu. Odezvou jsou brány například stisky příslušných tlačítek, v tomto příkladu je to odezva na stisknutí tlačítka *start*, nebo případně jak ukazuje úsek programu výběr výpočetní metody. Aby bylo možné pracovat se vstupními daty je využito příkazu *get*, který zajišťuje vlastní získání daného parametru. Protože každý uicontrol objekt je při deklaraci specifikován jedinečným tzv. *Tagem*, neboli popiskem, musí být objekt vyhledán. K vyhledávání objektů slouží příkaz nazvaný *findobj*, s jehož pomocí jsou objekty nalezeny a tedy aktivní. S takovýmto aktivním prvkem se poté může libovolně pracovat.

Podobně funguje i příkaz *set*, který narozdíl od *get* umožňí parametry objektů měnit. V užitém příkladu příkaz *Enable* zakazuje objekt definovaný *Tagem 'přesnost'*.

Protože data zadaná uživateli do vytvořeného prostředí jsou textovými řetezci, musí být tyto řetězce převedeny tak, aby byly pro Matlab srozumitelné. K tomuto slouží příkaz *str2num*, přeložený jako převod textové, nebo číselné hodnoty na numerickou formu.

```
polynom = str2num(get(findobj('Tag','koeficienty'),'String'));
```

Podobným způsobem pracuje příkaz *str2mat*, který převede řetězec zadaných znaků na matici. Jednotlivé prvky matice jsou odděleny čárkami. Příkaz slouží například pro výpis hodnot do *ListBoxu*.

```
vysledky=str2mat('METODA_MATLAB','Kořeny:',num2str(koreny)');
```

8.4 Nápověda k programu

Protože součástí každého programu je nápověda byla i pro tento program vytvořena ve formě jednoduchých HTML stránek za pomoci programu PsPad. Tento program je všetranným pomocníkem při programování nejen HTML stránek. Stránky nápovědy je možno vytvořit i upravovat v programovém prostředí Matlab za podmínky použití správného zápisu, neboli syntaxe. Možnosti jakým způsobem vytvořit nápovědu je mnoho, ale z hlediska nevelké složitosti nápovědy postačí k tvorbě html kód.

Nápověda je přístupná z vytvořeného menu programu v příslušné záložce. Předností nápovědy je snadné vyhledávání informací a praktické ukázky programu s příklady zadávání vstupních parametrů pro lepší pochopení problematiky.
Stručný obsah úvodní stránky nápovědy je:

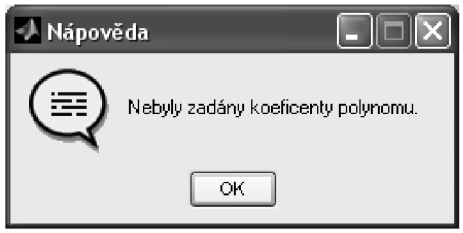
Použité metody – kapitola obsahuje teoretické poznatky o použitých metodách (o vlastnostech polynomů, Newtonově metodě tečen, metodě přírustku argumentu).

Ovládání programu – obsahuje celou stromovou strukturu vytvořeného menu programu, dále významy jednotlivých ovládácích prvků a také příklady používání.

Reference – do této části náleží seznam knižních titulů z nichž byli čerpány informace.

Odkazy – pomocné internetové odkazy na další méně významné teoretické podlady užitých metod, tyto metody nejsou nijak popsány v práci avšak jsou implementovány v aplikaci pro názorné srovnání výsledků existujích metod a metody nově vytvořené.

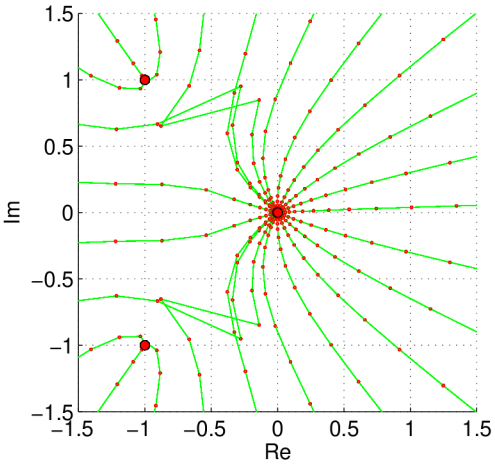
Program neobsahuje pouze nápovědu ve formě HTML stránek, ale také nedává uživateli přílišnou volnost v nastavování různých parametrů. Proto je prostředí vybaveno jednoduchými dialogy se zprávami, které informují uživatele o případných nesrovnalostech. Například špatně zadané, nebo naopak vůbec nezadané vstupní parametry. Dále je také program ošetřen deaktivací nepotřebných uicontrol objektů, které jsou jednoduše vypnuty a nemůže tudíž dojít k případným chybným hlášením a provedení nesprávné operace.



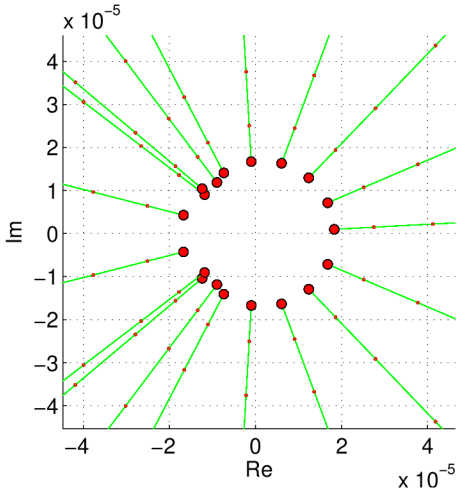
Obrázek 8.10: Dialogové okno použité v programu MPA.

9 DOSAŽENÉ VÝSLEDKY

Po vykonání všech operací uvnitř naprogramovaných funkcí poskytuje program všechny důležité informace mezi než patří výpis všech nalezených kořenů, násobnost jednotlivých kořenů a také porovnání dosažených výsledků s již existujícími metodami. Lokalizace vykreslená na obrázcích 9.1, 9.2 není grafickým výstupem programu, takovéto obrázky poskytuje lokalizační funkce bez použití uživatelského prostředí.



Obrázek 9.1: Lokalizace kořenů polynomu $x^5 + 2x^4 + 2x^3 = 0$.



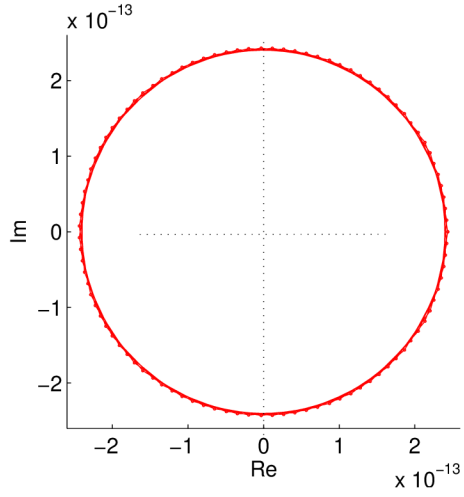
Obrázek 9.2: Lokalizace pro $\epsilon = 10^{-5}$.

Algoritmus metody přírustku argumentu aplikovaný na oblast D (kružnici), kde se předpokládá výskyt kořenů polynomu (2.1), vykreslí při jednom oběhu hranice C oblasti D, hladkou křivku v rovině obrazu ω. Změnšováním oblasti D kolem násobného kořene, který se nachází uvnitř oblasti křivka zachovává svůj hladký tvar mění se pouze měřítko. Křivka je zobrazena na obrázku 9.3. Protože z tohoto obrázku není patrný počet oběhů kolem počátku soustavy souřadnic je na obrázku 9.4 zobrazen detail hranice C, z něhož je možné vypozorovat, že se jedná o pětinásobný kořen, nebo pět blízkých kořenů. Křivka je získána transformací kružnice s těmito parametry:

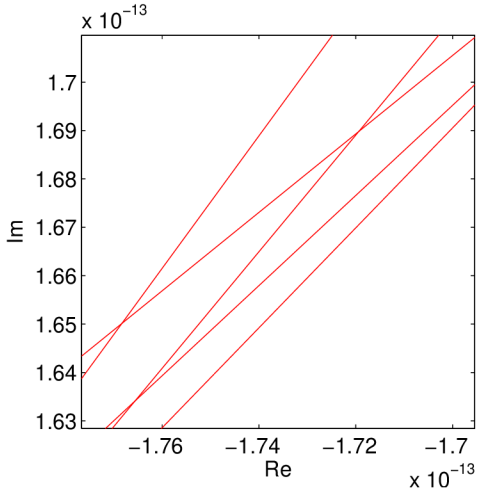
$$\begin{aligned} r &= 1,0001 + 0i \\ S &= 1 + 0i \end{aligned} \tag{9.1}$$

Křivka na obrázku. 9.5 je získána transformací menší oblasti, než je tomu na obrázku. 9.4. Konkrétně je tvořena body:

$$\begin{aligned} r &= 1,00097 + 0i \\ S &= 1 + 0i \end{aligned} \tag{9.2}$$

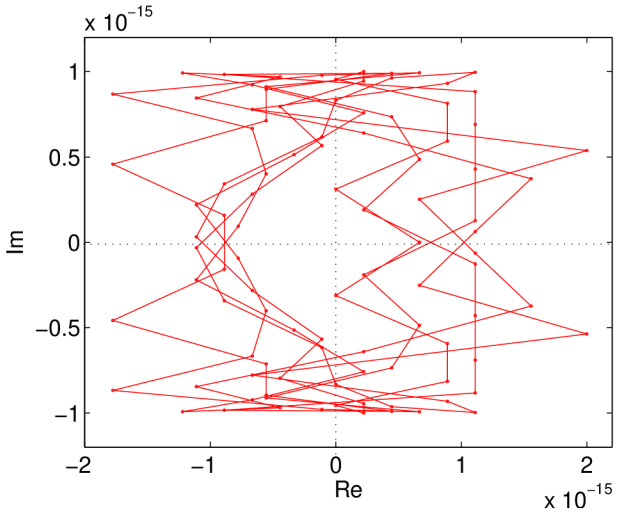


Obrázek 9.3: Křivka obrazu daná polynomem (2.1).



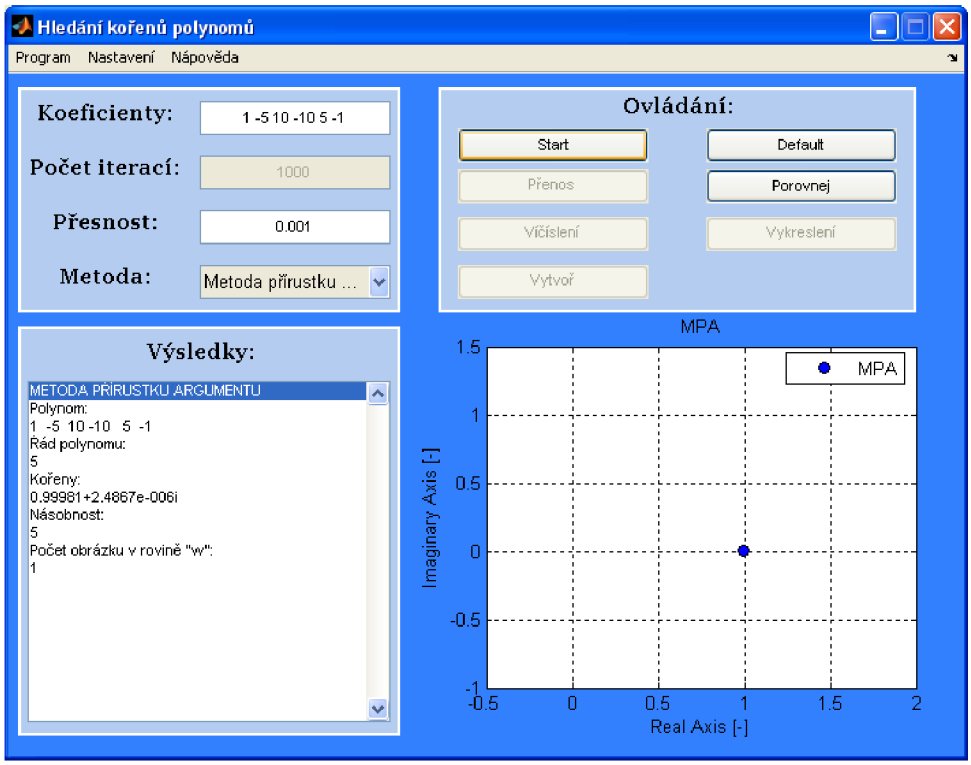
Obrázek 9.4: Detail křivky obrazu dané polynomem (2.1).

Je patrné, že křivka už není hladká. Stále je ale možné vypočítat počet oběhů křivky kolem počátku soustavy souřadnic. Špatného záznamu počtu kořenů bude dosaženo v případě, kdy se hranice zmenší ještě více než u (9.2). V takovém případě bude počet zaznamenaných kořenů < 5. Hranice algoritmu pro polynom (2.1) je kružnice se středem v bodu $1 + 0i$ a poloměrem o velikosti $1,00096$.

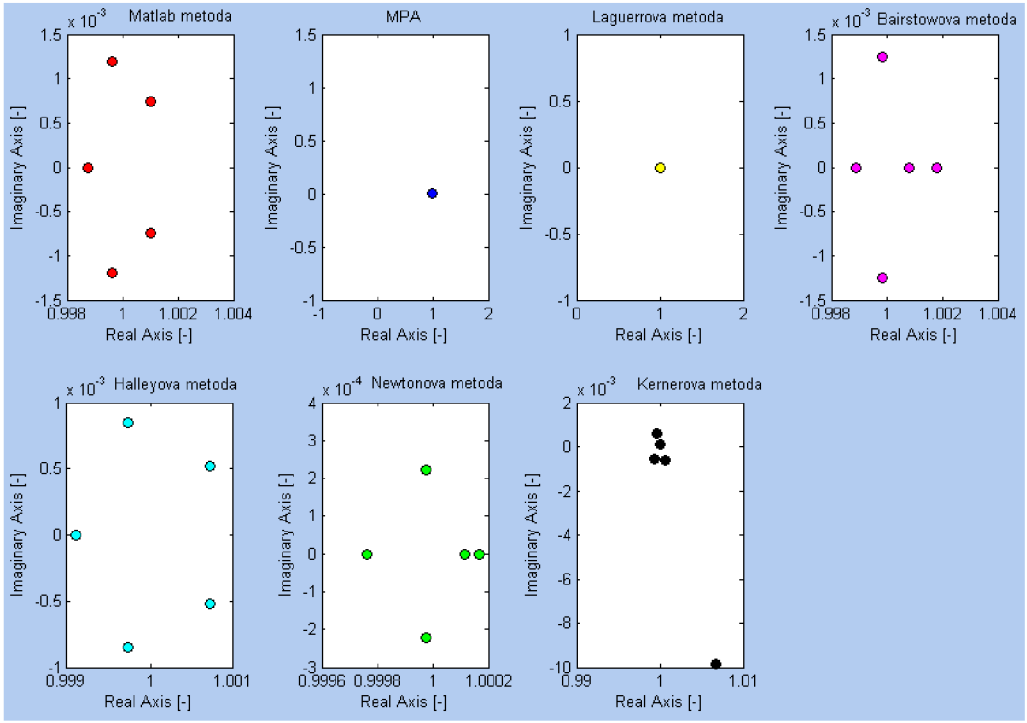


Obrázek 9.5: Křivka obrazu daná polynomem (2.1)pro skoro nedostačující poloměr.

Celkový vzhled grafického prostředí s poskytovanými informacemi je vyobrazen na obrázku 9.6. Porovnání výsledků pro polynom (2.1) různými metodami ukazuje další obrázek 9.7. Z něho vyplývá, že shodných výsledků dosahuje nově vytvořená metoda přírustku argumentu a také Laguerrova metoda. Ostatní použité metody naleznou shluk blízkých kořenů.



Obrázek 9.6: Okno navrženého GUI.



Obrázek 9.7: Porovnání výsledků jednotlivých metod pro polynom (2.1).

10 ZÁVĚR

Na základě poznatků o metodě přírustku argumentu a Newton-Raphsonově metodě tečen, byly vytvořeny nové funkce pro lokalizaci kořenů polynomů v programovém prostředí Matlab. Tělo programu se skládá z dvou hlavních funkcí, které se jednak starají o lokalizaci všech kořenů polynomu a také o ověřování násobností všech nalezených kořenů.

V porovnání s výsledky existujících metod aplikovaných do programu jsou dosažené výsledky více než přesvědčivé. Z celkem sedmi použitých metod dosahuje MPA nejlepších výsledků spolu s Laguerrovou metodou, která byla aplikována v programovém prostředí Matlab v jeho počátečních verzích. Velkou předností MPA oproti ostatním metodám je přesnější lokalizace násobných kořenů polynomů, což byl také předpoklad před započetím práce. Nevýhodou navržené metody je pouze delší výpočetní doba způsobená matematickými operacemi, které mají zpřesnit výsledky. Testování proběhlo na několika různých reálných polynomech (s násobnými, nenásobnými, komplexními kořeny, nebo na Wilkinsonových polynomech).

Bylo také vytvořeno uživatelské prostředí s různými možnostmi nastavení a výpočtů týkajících se polynomů. K programu byla naprogramována nápověda ve formě jednoduchých HTML stránek obsahující některé teoretické poznatky, ovládání, popis programu a některé další důležité informace. Nápověda je přístupná z okna grafického prostředí.

Práce byla také prezentována na studentské soutěži EEICT 2008 v kategorii Teoretická elektrotechnika, fyzika a matematika.

Díky této práci jsem získal ucelený přehled o využití matematiky aplikované v teoretické elektrotechnice a seznámil se velice dobře s možnostmi programového prostředí Matlab.

REFERENCE

[1] A.Ralston: *Základy numerické matematiky.* Academica, Praha, 1973.

[2] I.G.Aramanovic, G.L. Lunc, L.E. Elsgolc: *Funkcie komplexnej premenej, operátorový počet, teoria stability.* Alfa, SNTL, 1973.

[3] K. Hájek, J. Sedláček: *Kmitočtové filtry.* , BEN, 2002

[4] S. Míka: *Nemerické metody algebry.* , SNTL, 1985.

[5] J. Diblík, P. Sadovský: *Využití principu přírustku argumentu pro hledání kořene polynomu.* , 1. mezinárodní matematický workshop, ECON, 2003

[6] J. Diblík, P. Sadovský: *Principle of argument increment for searching polynomial roots.* , Radioelektronika 2003, BUT, 2003.

[7] P. Vavřín, P. Jura: *Systémy, procesy a signály 2.* , VUTIUM, 1999.

[8] T. Dostál: *Teorie elektronických obvodů.* , VUTIUM.

[9] P. Fuchs, V. Krupková: *Matematika 1.* , VUTIUM.

[10] B. Fajmon, I. Růžičková: *Matematika 3.* , VUTIUM.

[11] T. Dostál, Z. Kolka: *Analogové elektronické obvody počítačová cvičení.* , Brno, VUTIUM, 1.9.2003.

[12] K. Zaplatílek, B. Doňar: *Matlab, tvorba uživatelských aplikací.*, Praha, BEN, 2005.

[13] Analýza obvodů:. *informace o analýzách obvodů.* , WWW: <http://www.fs.vsb.cz/books/Analyza/>

[14] Princip argumentu:. *odvození platnosti vztahů principu argumetu.* , WWW: <http://nightmare.sh.cvut.cz/~sailor/>

[15] Matlab teorie řízení:. *Stručný manuál k Matlabu, Teorie řízení.* , WWW: <http://www.fm.vslib.cz/~krt/krt_cz/vyuka/text/matlab/html.htm>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

a_n koeficienty polynomu

$\delta(t)$ Diracův impuls

ϵ přesnost výpočtu

$f(x)$ definice funkce

$f^{'}(x)$ derivace funkce *f(x)*

$h(t)$ impulsní charakteristika

j komplexní jednotka

n stupeň polynomu

p Laplaceův operátor

P_n polynom n-tého stupně

R obor reálných čísel

$u(t)$ vstup systému

ω úhlová frekvence, nebo komplexní rovina

$y(t)$ výstup systému

x_0 kořen polynomu, nebo počáteční bod aproximace

z komplexní rovina

Z obor celých čísel

GUI grafické uživatelské prostředí

GUIDE grafické uživatelské vývojové prostředí

HTML značkovací jazyk pro vytváření dokumentů s hypertextovými odkazy

MPA metoda přírustku argumentu

MPI metoda půlení interavalu

RO regulační obvod

SDS spojité dynamické systémy

SEZNAM PŘÍLOH

A VYTVOŘENÉ M-FILE A HTML SOUBORY

A.1 m-file soubory

kresli-koreny.m – vykresování kořenů

kruhnwt15.m – lokalizační funkce

mpa.m – uživatelské prostředí

porovnej.m – porovnání výsledků jednotlivých metod

simulace.m – tvorba chrakteristik SDS

to02.m – ověření násobnosti

ulozpdf.m – možnost ukládání figure jako .pdf

vymaz.m – defaultní stav GUI

A.2 html soubory

informace.html – doplňující informace

moznosti.html – popis funkcí programu

mpa.html – základy metody přírustku argumentu

napoveda.html – hlavní menu nápovědy

newton.html – poznatky o Newtonově metodě tečen

polynomy.html – informace o polynomech

popis.html – popis jednotlivých částí programu

struktura.html – stromová struktura programu

vykreslovani.html – prezentování výsledků programu

B VLASTNOSTI GUI

B.1 Význam objektů v hiearchii

Axes – systém pro zobrazování 2D, 3D grafů v okně Figure

Figure – okno, ve kterém se zobrazuje grafika a uživatelská data

Image – 2D bitmapový obrázek

Light – zdroj osvětlení obrázku

Line – objekt čáry (využití při vykreslování grafů)

Patch – vyplněný polygon s hranami

Rect – 2D tvar s možností změny od čtvercového po oválný

Root – vrchol hiearchie grafických objektů, odpovídá obrazovce počítače

Surface – 3D prezentace maticových dat

Text – psaní textových řetězců v rámci definované grafiky

Uicontrol – uživatelské grafické rozhraní (vykonávání funkcí na reakce uživatele

Uimenu – Definované menu v okně Figure

Uicontextmenu – popupmenu, které je možné využít kdekoliv v rámci Root

B.2 Přehled použitých Uicontrol objektů

Axes – zobrazení a editace grafů, jeden ze základních výstupů aplikací

Edit text – tento objekt umožňuje vkládat nebo modifikovat textový řetězec

Figure – okno obsahující GUI nově naprogramované nebo vytvořené za pomoci průvodce systému Matlab

List box – zobrazení výsledků jako strukturovaný seznam(výsledná matice obsahuje seznam řetězců)

Popup menu – možnost výběru jedné položky z nabízeného seznamu

Push button – jednostavové tlačítko, příslušná akce je vygenerována jeho stiskem, uvolněním se vrací do původní nestlačené polohy

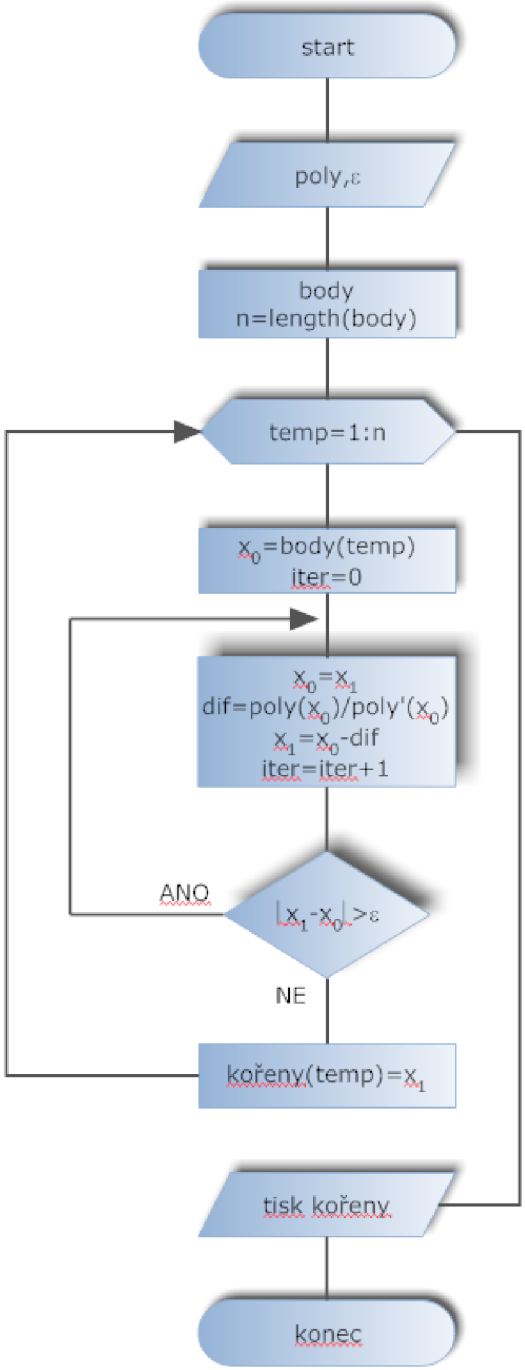
Static text – hlavní využití při popisu grafických objektů

C STRUKTURA PROGRAMU

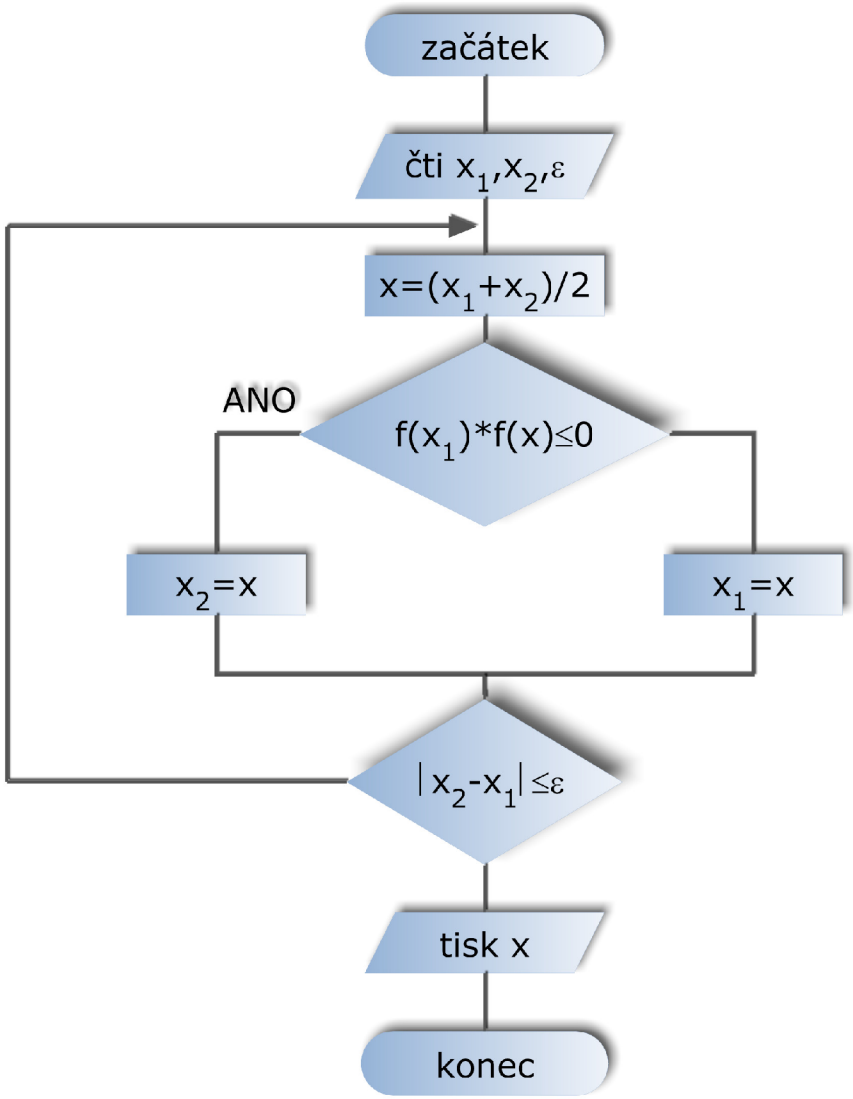
- Program
  - Otevřít
  - Uložit
  - Konec
- Nastavení
  - Výpočet
    - Kořeny polynomu
    - Přenosová funkce
    - Vytvoření polynomu
    - Vyčíslení polynomu
    - Vykreslení polynomu
  - Vykreslování
    - Hold On
    - Hold Off
    - Legenda
    - Mřížka
      - Zapnuta
      - Vypnuta
      - XGrid zapnuta
      - YGrid zapnuta
  - Vymazat
    - Vše
    - Editovací pole
    - Okno výsledky
    - Okno grafu
- Nápověda
  - Návod
  - O aplikaci

D VÝVOJOVÉ DIAGRAMY

D.1 Newtonova metoda tečen



D.2 Metoda půlení intervalu



D.3 Metoda Regula falsi