VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PARALELNÍ VÝPOČETNÍ ARCHITEKTURY ZALOŽENÉ NA NUMERICKÉ INTEGRACI

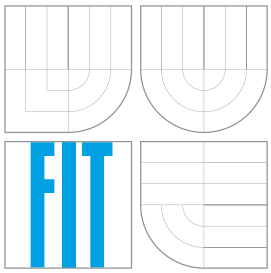
DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE
AUTHOR

Ing. MICHAL KRAUS

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PARALELNÍ VÝPOČETNÍ ARCHITEKTURY ZALOŽENÉ NA NUMERICKÉ INTEGRACI

PARALLEL COMPUTER SYSTEMS BASED ON NUMERICAL INTEGRATIONS

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE — Ing. MICHAL KRAUS
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JIŘÍ KUNOVSKÝ, CSc.
SUPERVISOR

BRNO 2013

Abstrakt

Předkládaná práce se zabývá simulací spojitých systémů popsaných soustavou diferenciálních rovnic nebo blokového diagramu. Zcela běžné je numerické řešení diferenciálních rovnic a používání simulačních programových celků (Matlab, Maple, TKSL). V této práci použitou numerickou metodou pro řešení diferenciálních rovnic je metoda Taylorovy řady. Bylo dokázáno, že metoda dosahuje velké přesnosti a rychlosti a nabízí možnost paralelního provádění a tím další urychlení výpočtu. Hlavní část práce obsahuje popis návrhu a realizace tohoto specializovaného paralelního systému v několika variantách a jejich porovnání.

Abstract

This thesis deals with continuous system simulation. The systems can be described by system of differential equations or block diagram. Differential equations are usually solved by numerical methods that are integrated into simulation software such as Matlab, Maple or TKSL. Taylor series method has been used for numerical solutions of differential equations. The presented method has been proved to be both very accurate and fast and also procesed in parallel systems. The aim of the thesis is to design, implement and compare a few versions of the parallel system.

Klíčová slova

numerická integrace, Taylorova řada, paralelní systém, propojovací sítě, integrátor

Keywords

numerical integration, Taylor series, parallel system, interconnection networks, integrator

Citace

Michal Kraus: Paralelní výpočetní architektury založené na numerické integraci, disertační práce, Brno, FIT VUT v Brně, 2013

Paralelní výpočetní architektury založené na numerické integraci

Prohlášení

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

.....
Michal Kraus
2. května 2013

Poděkování

Chtěl bych poděkovat svému školiteli doc. Ing. Jiřímu Kunovskému, CSc. za jeho podporu a vedení během mého studia, za cenné rady, připomínky a komentáře k mé práci. Můj dík patří také přítelkyni Petře, rodině a všem ostatním, kteří mě po dobu studia jakkoliv podporovali a bez nichž by tato práce nemohla vzniknout. Tato práce byla podporována MŠMT v rámci grantu MSM0021630528 - Výzkum informačních technologií z hlediska bezpečnosti a projektem FR2681/2010/G1 - Modernizace laboratorního přípravku do předmětu Prvky počítačů. Díky mezinárodnímu stipendijnímu programu Aktion jsem byl na roční stáži na TU Wien ve školním roce 2009/2010. Na základě tohoto pobytu vznikla dlouholetá spolupráce, která pokračuje dalšími výzkumnými projekty.

Obsah

Kapitola 1

Úvod

Přestože výpočetní kapacity sekvenčních počítačů neustále rostou, vždy se najdou náročné aplikace, pro které není dostupný výkon dostačující. Myšlenka znásobení stávajícího výkonu paralelizací je prostá a je stará v podstatě jako výpočetní technika sama. Myšlenku lze formulovat takto: Sdružme několik počítačů a rozdělme výpočetní zátěž mezi ně. Budeme moci zpracovat větší úlohy a výpočet proběhne rychleji. Jinými slovy: Paralelní zpracování slibuje zvýšení výkonu, kapacity, poměru výkonu a ceny, spolehlivosti apod.

Základní problémy vědy a inženýrství s širokými ekonomickými a vědeckými dopady na naší civilizaci vyžadují následující vysoce náročné výpočty (High Performance Computing - HPC):

- předpovídání počasí, modelování atmosféry, moří a zemětřesení
- modelování v astronomii, kosmologii, biochemii a genetice
- návrhy a testy složitých inženýrských systémů (letadla, auta, elektrárny)
- rozpoznání lidské řeči, modelováni lidské inteligence, robotika
- a mnoho dalších

Modelování a simulace reálných systémů hraje významnou roli ve všech odvětvích dnešní společnosti. Pro simulaci reálných systémů reálného světa se používají dva odlišné přístupy: spojitý a diskrétní. O tom, který přístup je vhodnější, rozhoduje úroveň abstrakce, která se při simulaci použije. Odpovídající výběr popisu modelu samozřejmě závisí jak na účelu modelu, tak na úrovni našich znalostí o modelovaném systému.

1.1 Motivace, cíle disertační práce

Předkládaná disertační práce se zabývá simulací spojitých systémů. Spojité systémy se popisují soustavou diferenciálních a algebraických rovnic, velmi přirozenou formou je rovněž popis spojitých systémů pomocí blokového diagramu (zcela typická je tato situace v teorii řízení). Blokový diagram je chápán jako paralelně pracující systém. Při výpočtu se odpovídající blokový diagram převede na soustavu obyčejných diferenciálních rovnic, které se řeší v zásadě analyticky nebo numericky. Analytické řešení soustav diferenciálních rovnic je omezeno jen na určitou omezenou třídu úloh a je velmi komplikované. Zcela běžné je proto numerické řešení diferenciálních rovnic a používání simulačních programových celků (Matlab [Mat12], Maple [Map12], TKSL [Kun94]).

Nejčastěji se používají metody Runge-Kutta, Eulerova metoda, nebo vícekrokové metody typu prediktor-korektor. Zde prezentovaná metoda využívající Taylorovy řady se od běžných metod liší možností výpočtu vyšších derivací a jejich následným využitím při výpočtu. Bližší seznámení s metodou Taylorovy řady je v [Kun94]. Srovnání přesnosti a rychlosti výpočtu za pomocí této metody s ostatními používanými metodami bylo obsahem několik vědeckých článků publikovaných na konferencích a v časopisech, např [1] a [17]. Z rozboru prezentovaným také v této práci je patrné, že metoda nabízí možnost paralelního provádění výpočtu a tím další urychlení.

Hlavní cíle předložené práce shrnu do několika následujících bodů:

- Vymezení typu diferenciálních rovnic, které je možno řešit metodou Taylorovy řady a následně tedy i navrženým paralelním systémem
- Srovnání metody Taylorovy řady s numerickými metodami světových standardů (Matlab, Maple)
- Návrh specializovaného paralelního systému určeného pro numerické řešení diferenciálních rovnic
- Rozbor a srovnání variant integrátorů a celých paralelních systémů

1.2 Struktura práce

Struktura je volena tak, aby odrážela současný stav vědění v tématu disertační práce a obsahovala všechny důležité informace popisující realizovanou výzkumnou činnost a její výsledky.

Důležitou náplní této práce je numerická integrace. Základními rysy a vlastnostmi metod numerické integrace se zabývá kapitola 2. Jsou zde uvedeny základní pojmy numerické integrace i se stručným popisem jednokrokových a vícekrokových numerických integračních metod. Hlavně je však uvedena nově pojatá metoda Taylorovy řady (v sočasné době implementovaná v simulačním jazyce TKSL) a porovnána s používanými metodami Runge-Kutta. Metoda Taylorovy řady se vyznačuje nejen vysokou přesností výpočtu, ale především vysokým stupněm paralelismu. Výpočetní operace (numerické integrace) jsou při použití moderní metody Taylorovy řady na sobě ve speciálních případech nezávislé a mohou být realizovány v oddělených paralelně pracujících procesorech.

Následující kapitoly jsou už zaměřeny na samotný specializovaný paralelní systém. Základní verze paralelní výpočetní architektury založené na numerické integraci obsahuje výpočetní aritmeticko-logické jednotky, propojovací systém a řídící jednotku. Možné varianty výpočetní architektury vycházejí ze tří verzí aritmeticko-logických jednotek (tří verzí numerických integrátorů). Kapitola 3 obsahuje popis návrhu celého výukového systému. Konkrétně návrh tří typů aritmeticko-logických jednotek, řídící jednotky a propojovacího systému. Kapitola 4 obsahuje popis realizace jednotlivých prvků specializovaného paralelního systému a porovnání jednotlivých variant.

Kapitola 5 shrnuje dosažené výsledky disertační práce, hodnotí splnění cílů a představuje možnosti dalšího výzkumu v této oblasti.

Kapitola 2

Numerické řešení diferenciálních rovnic

Nalezení analytického řešení rozsáhlých soustav diferenciálních rovnic je složité, mnohdy nemožné. V současné době se s rozmachem výpočetní techniky čím dál častěji využívá numerického řešení. Zaměřme se tedy na problém numerického řešení obyčejných diferenciálních rovnic s počáteční podmínkou.

Počáteční problém pro ODR.
Obyčejná diferenciální rovnice (ODR) se nazývá rovnice *n-tého řádu* (ODRn), jestliže neznámá funkce je funkcí jedné proměnné a její nejvyšší derivace neznámé funkce je *n*-tého řádu. V práci se budeme zabývat především diferenciálními rovnicemi prvního řádu. Popis jednotlivých typů obyčejných diferenciálních rovnic a způsobů jejich řešení obsahuje [Šva99, DB91].

Obecný tvar diferenciální rovnice prvního řádu je

$$g(t, y(t), y'(t)) = 0. \tag{2.1}$$

Předpokládejme, že rovnici (2.1) lze také vyjádřit explicitně ve tvaru

$$y'(t) = f(t, y(t)). \tag{2.2}$$

Obecné řešení diferenciální rovnice obsahuje integrační konstantu, která může nabývat libovolné hodnoty. Proto k jednoznačnému určení $y(t)$ musíme ještě doplnit hodnotu funkce v určitém bodě $t = t_0$, tedy *počáteční podmínku*

$$y(t_0) = y_0. \tag{2.3}$$

Rovnice (2.2) spolu s počáteční podmínkou (2.3) se nazývá *počáteční úloha*, nebo také *Cauchyova úloha*.

Budeme předpokládat, že každá Cauchyova úloha, o jejímž numerickém řešení budeme uvažovat, má jediné řešení. Je dobře známo, že tato podmínka **existence a jednoznačnosti řešení** je splněna, když je funkce $f(t, y(t))$ spojitá, ohraničená a splňuje Lipschitzovu podmínku.

Předcházející část obsahující teoretický popis diferenciálních rovnic byla převzata z [Šát12].

Za numerické řešení obyčejné diferenciální rovnice s počáteční podmínkou (2.4)

$$y' = f(t, y), \qquad y(t_0) = y_0 \tag{2.4}$$

je považována sekvence hodnot (2.5)

$$[y(t_0) = y_0], \quad [y(t_1) = y_1], \quad \dots \quad [y(t_N) = y_N]. \tag{2.5}$$

Tato čísla aproximují hodnoty přesného analytického řešení $y(t_0), y(t_1), \dots, y(t_N)$ v bodech $t_0, t_1, \dots, t_N$. Obvykle volíme ekvidistantní síť, tj. $t_i - t_{i-1} = h$; $i = 0, 1, \dots, N$. Číslo h se nazývá integrační krok. Hodnoty funkce mezi zvolenými body lze určit buď interpolací z okolních vypočtených bodů, nebo opětovnou aplikací numerické metody s použitím menšího integračního kroku.

Výpočet probíhá iteračně. Numerické řešení y_i v bodě t_i se počítá z hodnoty předešlého numerického řešení y_{i-1}. To platí pro *jednokrokové metody*. Existují i *vícekrokové metody*, které k výpočtu aktuálního řešení y_i používají k předešlých výsledků $y_{i-1}, y_{i-2}, \dots, y_{i-k}$. Při využití vícekrokových metod je problematický postup v prvních krocích výpočtu, kdy ještě nemáme k dispozici dostatečný počet předcházejících hodnot. Pro zahájení výpočtu je tedy nutné použít některou z jednokrokových metod.

Diferenciální rovnice vyšších řádů se musí pro numerické řešení převést na soustavu obyčejných diferenciálních rovnic prvního řádu. K realizaci tohoto převodu může být použita:

- metoda snižování řádu derivace - jednodušší, ale vyžaduje, aby na pravé straně rovnice nebyly derivace vstupu
- metoda postupné integrace - zvládne i rovnice s derivacemi vstupu na pravé straně

Obě metody využívají úprav rovnice a zavádění pomocných proměnných. Jsou použitelné i na soustavy rovnic vyššího řádu - stačí postup opakovat pro každou zadanou rovnici.

2.1 Numerické metody pro řešení diferenciálních rovnic

Bylo již publikováno veliké množství numerických metod řešení diferenciálních rovnic navzájem se lišících především ve dvou hlavních kritériích - *přesnosti* a *rychlosti*. Tato kritéria stojí proti sobě. Vždy je jedno kritérium potlačeno na úkor druhého. Popis numerických metod je v [KDJ05]. V této práci uvádím jen popis několika základních metod nutných pro definování principu výpočtu.

2.1.1 Taylorova řada

Taylorova řada je základní jednokroková metoda a vyjadřuje se zápisem:

$$y_{i+1} = y_i + h \cdot f^{[1]}(t_i, y_i) + \frac{h^2}{2!} \cdot f^{[2]}(t_i, y_i) + \dots + \frac{h^p}{p!} \cdot f^{[p]}(t_i, y_i) \tag{2.6}$$

h je integrační krok. Tato metoda umožňuje určit přesnost výpočtu. Výpočet konkrétní hodnoty y_{i+1} je iteračně prováděn až po dosažení zadané přesnosti výpočtu, kdy rozdíl dvou po sobě následujících hodnot členů Taylorovy řady je menší než požadovaná přesnost.

Hlavní problém spojený s použitím Taylorovy řady spočívá v nutnosti použití vyšších derivací. Aplikace Taylorovy řady na numerické řešení obyčejných diferenciálních rovnic jsou popsány např. v [Gib60, BWZ72].

2.1.2 Eulerova metoda

Nejjednodušší jednokroková Eulerova metoda využívá pouze první dva členy Taylorovy řady:

$$y_{i+1} = y_i + h \cdot f^{[1]}(t_i, y_i) \tag{2.7}$$

Z hlediska geometrie vyjadřuje pravá strana rovnici přímky, kde $f^{[1]}(t_i, y_i)$ je směrnice tečny v bodě t_i.

Metoda se používá pro dostatečně malé h.

2.1.3 Metody Runge-Kutta

Vyšší přesnost a rychlost výpočtu ve srovnání s Eulerovou metodou lze získat metodami Runge-Kutta. Výpočet provádíme i mezi uzly y_i a y_{i+1}. Výsledný přírůstek najdeme jako váhový průměr vypočtených hodnot a jejich počet nám udává řád metody.

Obecný tvar jednokrokových metod Runge-Kutta je

$$y_{i+1} = y_i + h \cdot (w_1 k_1 + \cdots + w_s k_s) \tag{2.8}$$

- $k_1 = f(t_i, y_i)$
- $k_i = f(t_i + \alpha_l h, y_i + h \sum_{j=1}^{l-1} \beta_{lj} k_j), \quad i = 2, \ldots, s$
- konstanty w_n, α_n a β_{lj} jsou konstanty volené podle řádu metody tak, aby získané řešení souhlasilo s Taylorovou řadou v bodě t_{i+1}.

Metoda Runge-Kutta 2. řádu je používána ve dvou variantách (označované jako lichoběžníková (2.9) a zlepšený Euler (2.10)). Jejich obecný tvar je:

$$\begin{aligned} y_{i+1} &= y_i + h \cdot (\frac{1}{2} k_1 + \frac{1}{2} k_2) \\ k_1 &= f(t_i, y_i) \\ k_2 &= f(t_{i+1} + h, y_i + h \cdot k_1) \end{aligned} \tag{2.9}$$

$$\begin{aligned} y_{i+1} &= y_i + k_2 \\ k_1 &= f(t_i, y_i) \\ k_2 &= f(t_{i+1} + \frac{1}{2} h, y_i + \frac{1}{2} h \cdot k_1) \end{aligned} \tag{2.10}$$

Nejvíce rozšířenou a používanou metodou je Runge-Kutta 4. řádu. Její obecný tvar je (2.11)

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{6} h \cdot (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4) \\ k_1 &= f(t_i, y_i) \\ k_2 &= f(t_{i+1} + \frac{1}{2} h, y_i + \frac{1}{2} h \cdot k_1) \\ k_3 &= f(t_{i+1} + \frac{1}{2} h, y_i + \frac{1}{2} h \cdot k_2) \\ k_4 &= f(t_{i+1} + h, y_i + h \cdot k_3) \end{aligned} \tag{2.11}$$

Více informací o metodách Rungu-Kutta je v [But08].

2.1.4 Metody Adams-Bashforth

Jedná se o první významnou numerickou metodu pro řešení obyčejných diferenciálních rovnic. Byla vyvinuta již v 19. století [But00] a dodnes tvoří významnou část ve většině moderních softwarových nástrojů. Jedná se o vícekrokovou metodu. Např. metoda Adams-Bashforth počítá následující hodnotu ze čtyř předchozích hodnot podle vzorce

$$y_{i+1} = y_i + \frac{h}{24} \cdot (55y_i - 59y_{i-1} + 37y_{i-2} - 9y_{i-3}) \tag{2.12}$$

Problém nastává při spuštění výpočtu. Metoda není samostartující a je nutné použít jednokrokovou metodu.

2.2 Metoda Taylorovy řady

Analytické počítání členů Taylorovy řady (především odvozování vyšších derivací) bylo považováno v mnoha případech za příliš komplikované. V mnoha moderních knihách o numerické a aplikované matematice je uvedeno, že Taylorova řada je z teoretického a koncepčního hlediska velice zajímavá, avšak může být použita jen v určitých speciálních případech. Obecné použití integračního algoritmu je příliš složité. Předkládaná práce se zabývá běžnými "technickými problémy". Pro jejich řešení lze Taylorovu řadu úspěšně využít [Mik00].

Důležitou součástí výpočtu je automatické nastavení řádu metody. To znamená, že se použije tolik členů Taylorovy řady, kolik je potřeba k dosažení požadované přesnost výpočtu. Metoda Taylorovy řady má velmi příznivé paralelní vlastnosti. Většina výpočetních operací je vzájemně nezávislých, takže mohou být prováděny paralelně v oddělených procesorech paralelního výpočetního systému. Nezbytnou součástí metody je automatická transformace zadání. Původně zadaná soustava nelineárních diferenciálních rovnic se automaticky transformuje na polynomiální tvar, t.j. na tvar, u kterého lze snadno rekurentně vypočítat jednotlivé členy Taylorovy řady. Tato transformace je představena v podkapitole 2.2.3.

I když nebyla metoda Taylorovy řady v literatuře velmi preferovaná kvůli nutnosti použití vyšších derivací, experimentální výpočty ukázaly a teoretické analýzy dokázaly, že přesnost a stabilita metody Taylorovy řady předčí běžně používané explicitní metody pro numerické řešení diferenciálních rovnic. Tato srovnání jsou také obsahem několika vědeckých článků.

V článku [10] je ukázáno řešení homogenní diferenciální rovnice s konstantními koeficienty. Na tomto řešení je předvedena extrémní přesnost a rychlost výpočtu pomocí Taylorovy řady. Jsou zde porovnány různé šířky výpočetní aritmetiky a zobrazeny závislosti na změně použitého integračního kroku a počtu členů Taylorovy řady.

Článek [17] představuje využití metody Taylorovy řady k výpočtu diferenciálních rovnic. Zaměřuje se na speciální případy, které vyžadují vysokou přesnost výpočtu. Porovnává numerické řešení diferenciálních rovnic pomocí TKSL a Matlabu.

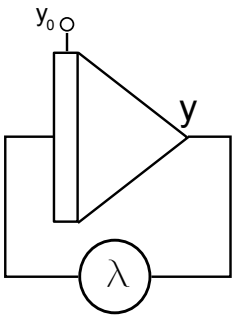
Článek [1] zahrnuje také řešení problémů, které mohou být transformovány na řešení systému diferenciálních rovnic. V článku je Taylorova řada aplikována na konkrétní reálný model torzní hřídele automobilu a diskutována stabilita a konvergence. Dále je Taylorova řada srovnána s metodami, které se využívají v programu Matlab/Simulink. Článek vznikl ve spolupráci s Technickou univerzitou ve Vídni během mého stipendijního pobytu.

2.2.1 Použití Taylorovy řady

Předkládaná nová specializovaná metoda Taylorovy řady [Kun94] pro numerické řešení obyčejných diferenciálních rovnic byla vytvořena na základě analýzy nejjednodušší homogenní lineární diferenciální rovnice 1. řádu s konstantními koeficienty - tzv. *Dahlquistův problém* [HW96].

$$y' = \lambda \cdot y, \qquad y(0) = y_0 \tag{2.13}$$

Odpovídající blokové řešení s využitím známého symbolu integrátoru a násobící konstanty je znázorněno na obrázku 2.1:



Obrázek 2.1: Blokové znázornění - Dahlquistův problém

V následujícím textu se rovnice (2.6)–(2.11) využívající zápis $y' = f(t, y)$ modifikují na $y' = f(y), y(0) = y_0$. Zápis rovnice (2.13) umožňuje jednoduše definovat pro výpočet prvního kroku numerického řešení následující algoritmy výpočtu (pro $\lambda = 1$):

- při aplikaci Eulerovy metody (rovnice 2.7):

$$\begin{aligned} y_1 &= y_0 + DY1_0 \\ DY1_0 &= h \cdot y_0 \end{aligned} \tag{2.14}$$

- při aplikaci metody Runge-Kutta 2. řádu (rovnice 2.10):

$$\begin{aligned} y_1 &= y_0 + k_2 \\ k_1 &= h \cdot y_0 \\ k_2 &= h \cdot (y_0 + \frac{k_1}{2}) \end{aligned} \tag{2.15}$$

- při aplikaci standardní metody Runge-Kutta 4. řádu (rovnice 2.11):

$$\begin{aligned} y_1 &= y_0 + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \\ k_1 &= h \cdot y_0 \\ k_2 &= h \cdot (y_0 + \frac{k_1}{2}) \\ k_3 &= h \cdot (y_0 + \frac{k_2}{2}) \\ k_4 &= h \cdot (y_0 + \frac{k_3}{2}) \end{aligned} \tag{2.16}$$

Správnost definovaných algoritmů (rovnice 2.14–2.16) (při numerickém řešení rovnice 2.13) lze potvrdit úpravou vztahů pro výpočet hodnoty y_1:

- při aplikaci Eulerovy metody (po dosazení do rovnice 2.14) je

$$y_1 = y_0 \cdot (1 + h) \tag{2.17}$$

- při aplikaci metody Runge-Kutta 2. řádu (po dosazení do rovnice 2.15) je

$$y_1 = y_0 \cdot (1 + h + \frac{h^2}{2}) \tag{2.18}$$

- při aplikaci metody Runge-Kutta 4. řádu (po dosazení do rovnice do 2.16) je

$$y_1 = y_0 \cdot (1 + h + \frac{h^2}{2} + \frac{h^3}{3!} + \frac{h^4}{4!}) \tag{2.19}$$

Ze zápisu rovnic (2.17)–(2.19) je zřejmé, že se jedná o zcela charakteristický rozvoj funkce do exponenciální řady ($e^t = 1 + t/1! + t^2/2! + t^3/3! + \dots$). Tento výsledek bylo možno očekávat, protože známým analytickým řešením rovnice (2.13) je

$$y = e^t \tag{2.20}$$

Stejný výsledek zápisu první hodnoty numerického výpočtu y_1 rovnice (2.13) lze však získat i rozvojem funkce y (rovnice 2.20) do Taylorovy řady (rovnice 2.1.1). Podmínka platnosti Taylorovy řady (funkce $f(t)$ a její derivace musí být jednoznačné, konečné a spojité v intervalu mezi t a $t + h$) je splněna, protože platí

$$y' = y \qquad \text{resp.} \qquad y' = y'' = y''' = \cdots \tag{2.21}$$

- při aplikaci metody Taylorovy řady (po dosazení do rovnice 2.6) je

$$y_1 = y_0 + h \cdot y_0 + \frac{h^2}{2!} y_0 + \frac{h^3}{3!} y_0 + \cdots \tag{2.22}$$

tzn. skutečně je

$$y_1 = y_0 \cdot (1 + h + \frac{h^2}{2} + \frac{h^3}{3!} + \cdots)$$

Provede-li se v rovnici (2.22) označení

$$y_1 = y_0 + DY1_0 + DY2_0 + DY3_0 + \cdots \tag{2.23}$$

potom platí

$$DY1_0 = h \cdot y_0 \tag{2.24}$$

$$DY2_0 = \frac{h}{2} DY1_0 \tag{2.25}$$

$$DY3_0 = \frac{h}{3} DY2_0 \tag{2.26}$$

$$\vdots$$

$$DYp_0 = \frac{h}{p} DY(p-1)_0 \tag{2.27}$$

Lze tedy rovnice 2.23–2.27 považovat za algoritmus numerického výpočtu hodnoty y rovnice (2.13) metodou Taylorovy řady.

Prakticky se výpočet provede tak, že se ze známé počáteční podmínky y_0 stanoví (dle rovnice 2.24) výraz $DY1_0$ (tj. členy Taylorovy řady s první mocninou kroku h). Z této vypočítané hodnoty $DY1_0$ se stanoví (dle rovnice 2.25) výraz $DY2_0$ (tj. člen Taylorovy řady s druhou mocninou kroku h). Tímto postupem se pokračuje až do p-tého řádu.

Nová hodnota řešení y_1 (tj. hodnota prvního kroku výpočtu) se stanoví jako součet počáteční podmínky y_0 a dílčích přírůstků (členů Taylorovy řady) - rovnice (2.23).

Postup výpočtu metodou Taylorovy řady je velmi jednoduše algoritmizovatelný - z hlediska uživatele stačí definovat počáteční podmínku, zvolený integrační krok h a řád metody - nabízí se rovněž varianta výpočtu na "plnou přesnost", tzn. bude se provádět výpočet takovým řádem metody (s tolika členy Taylorovy řady), pokud se hodnota DYp_0 uplatní v součtu (rovnice 2.23).

2.2.2 Srovnání efektivnosti Taylorovy řady a metody Runge-Kutta

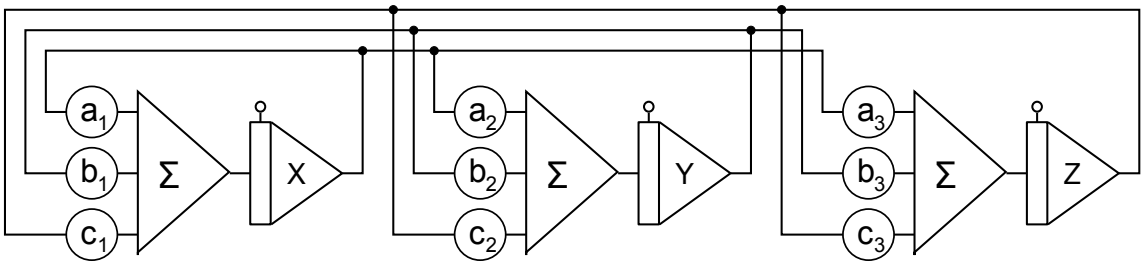
Postup numerického výpočtu rovnice (2.13) lze aplikovat na soustavu homogenních lineárních diferenciálních rovnic s konstantními koeficienty (např. pro soustavu rovnic (2.28)–(2.30))

$$x' = a_1 \cdot x + b_1 \cdot y + c_1 \cdot z \qquad x(0) = x_0 \tag{2.28}$$

$$y' = a_2 \cdot x + b_2 \cdot y + c_2 \cdot z \qquad y(0) = y_0 \tag{2.29}$$

$$z' = a_3 \cdot x + b_3 \cdot y + c_3 \cdot z \qquad z(0) = z_0 \tag{2.30}$$

Ze zápisu numerických řešení této soustavy (přesněji, z výpočtu jednoho kroku numerického řešení) homogenních lineárních diferenciálních rovnic s konstantními koeficienty vyplývají algoritmy paralelní spolupráce nezávislých procesorů (integrátorů). Odpovídající blokové schéma je na obrázku 2.2.



Obrázek 2.2: Blokové schéma pro řešení soustavy diferenciálních rovnic (2.28)–(2.30)

Cílem následující části je provést srovnání efektivnosti Taylorovy řady a metody Runge-Kutta při řešení soustavy homogenních diferenciálních rovnic 1. řádu (2.28)–(2.30).

Řešení soustavy (2.28)–(2.30) metodou Runge-Kutta 2. řádu

Obecný tvar pro metodu Runge-Kutta 2. řádu (2.9) lze pro první krok výpočtu soustavy (2.28)–(2.30) přepsat do tvaru:

$$x_1 = x_0 + \frac{1}{2}(k_{1,x} + k_{2,x}) \tag{2.31}$$

$$y_1 = y_0 + \frac{1}{2}(k_{1,y} + k_{2,y}) \tag{2.32}$$

$$z_1 = z_0 + \frac{1}{2}(k_{1,z} + k_{2,z}) \tag{2.33}$$

kde pro koeficienty k_1 platí:

$$k_{1,x} = h(a_1 \cdot x_0 + b_1 \cdot y_0 + c_1 \cdot z_0) \tag{2.34}$$

$$k_{1,y} = h(a_2 \cdot x_0 + b_2 \cdot y_0 + c_2 \cdot z_0) \tag{2.35}$$

$$k_{1,z} = h(a_3 \cdot x_0 + b_3 \cdot y_0 + c_3 \cdot z_0) \tag{2.36}$$

a pro koeficienty k_2:

$$k_{2,x} = h(a_1(x_0 + k_{1,x}) + b_1(y_0 + k_{1,y}) + c_1(z_0 + k_{1,z})) \tag{2.37}$$

$$k_{2,y} = h(a_2(x_0 + k_{1,x}) + b_2(y_0 + k_{1,y}) + c_2(z_0 + k_{1,z})) \tag{2.38}$$

$$k_{2,z} = h(a_3(x_0 + k_{1,x}) + b_3(y_0 + k_{1,y}) + c_3(z_0 + k_{1,z})) \tag{2.39}$$

Ze zápisu rovnic (2.31)–(2.39) je zřejmé, že požadované matematické operace v navzájem si odpovídajících rovnicích (tj. (2.31)–(2.33), (2.34)–(2.36) a (2.37)–(2.39)) jsou stejné. Mohou být tedy provedeny paralelně ve třech oddělených mikroprocesorech. Předpokládá se, že procesory provádí základní elementární operace (sčítání a násobení).

Základní posloupnost všech operací při řešení soustavy pro všechny paralelní procesory je názorně uvedena v tabulce 2.1.

Postupnými elementárními operacemi se získá k_1: konkrétně v procesoru X je $k_{1,x} = X_6$, v procesoru Y je $k_{1,y} = Y_6$ a v procesoru Z je $k_{1,z} = Z_6$. Dále následuje obdobným způsobem souběžně ve všech třech procesorech postupný výpočet k_2: v procesoru X je $k_{2,x} = X_{15}$, v procesoru Y je $k_{2,y} = Y_{15}$ a v procesoru Z je $k_{2,z} = Z_{15}$. Na závěr se z vypočtených hodnot k_1, k_2 a počátečních podmínek dopočítá výsledek $x_1 = X_{18}$, $y_1 = Y_{18}$ a $z_1 = Z_{18}$.

Řešení soustavy (2.28)–(2.30) metodou Taylorovy řady 2. řádu

Pokud použijeme označení analogicky s rovnicí (2.23), obdržíme soustavu pro numerický výpočet nových hodnot x_1, y_1 a z_1 ve tvaru (použijí se pouze 2 členy Taylorovy řady):

$$x_1 = x_0 + DX1_0 + DX2_0 \tag{2.40}$$

$$y_1 = y_0 + DY1_0 + DY2_0 \tag{2.41}$$

$$z_1 = z_0 + DZ1_0 + DZ2_0 \tag{2.42}$$

kde význam jednotlivých členů je:

$$DX1_0 = h \cdot (a_1 \cdot x_0 + b_1 \cdot y_0 + c_1 \cdot z_0) \tag{2.43}$$

$$DY1_0 = h \cdot (a_2 \cdot x_0 + b_2 \cdot y_0 + c_2 \cdot z_0) \tag{2.44}$$

$$DZ1_0 = h \cdot (a_3 \cdot x_0 + b_3 \cdot y_0 + c_3 \cdot z_0) \tag{2.45}$$

Tabulka 2.1: Posloupnost operací při řešení soutavy metodou Runge-Kutta 2. řádu

Pořadí	**Procesor X**	**Procesor Y**	**Procesor Z**
1	$X1 = a_1 \cdot x_0$	$Y1 = a_2 \cdot x_0$	$Z1 = a_3 \cdot x_0$
2	$X2 = b_1 \cdot y_0$	$Y2 = b_2 \cdot y_0$	$Z2 = b_3 \cdot y_0$
3	$X3 = c_1 \cdot z_0$	$Y3 = c_2 \cdot z_0$	$Z3 = c_3 \cdot z_0$
4	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$
5	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$
6	$X6 = h \cdot X5$	$Y6 = h \cdot Y5$	$Z6 = h \cdot Z5$
7	$X7 = x_0 + X6$	$Y7 = x_0 + X6$	$Z7 = x_0 + X6$
8	$X8 = y_0 + Y6$	$Y8 = y_0 + Y6$	$Z8 = y_0 + Y6$
9	$X9 = z_0 + Z6$	$Y9 = z_0 + Z6$	$Z9 = z_0 + Z6$
10	$X10 = a_1 \cdot X7$	$Y10 = a_2 \cdot Y7$	$Z10 = a_3 \cdot Z7$
11	$X11 = b_1 \cdot X8$	$Y11 = b_2 \cdot Y8$	$Z11 = b_3 \cdot Z8$
12	$X12 = c_1 \cdot X9$	$Y12 = c_2 \cdot Y9$	$Z12 = c_3 \cdot Z9$
13	$X13 = X10 + X11$	$Y13 = Y10 + Y11$	$Z13 = Z10 + Z11$
14	$X14 = X13 + X12$	$Y14 = Y13 + Y12$	$Z14 = Z13 + Z12$
15	$X15 = h \cdot X14$	$Y15 = h \cdot Y14$	$Z15 = h \cdot Z14$
16	$X16 = X6 + X15$	$Y16 = Y6 + Y15$	$Z16 = Z6 + Z15$
17	$X17 = 1/2 \cdot X16$	$Y17 = 1/2 \cdot Y16$	$Z17 = 1/2 \cdot Z16$
18	$X18 = x_0 + X17$	$Y18 = y_0 + Y17$	$Z18 = z_0 + Z17$

$$DX2_0 = \frac{h}{2} \cdot (a_1 \cdot DX1_0 + b_1 \cdot DY1_0 + c_1 \cdot DZ1_0) \tag{2.46}$$

$$DY2_0 = \frac{h}{2} \cdot (a_2 \cdot DX1_0 + b_2 \cdot DY1_0 + c_2 \cdot DZ1_0) \tag{2.47}$$

$$DZ2_0 = \frac{h}{2} \cdot (a_3 \cdot DX1_0 + b_3 \cdot DY1_0 + c_3 \cdot DZ1_0) \tag{2.48}$$

Ze zápisu rovnic (2.40)–(2.48) je zřejmé, že výpočet lze provést opět paralelně. Základní posloupnost operací při řešení soustavy pro všechny paralelní procesory je uvedena v tabulce 2.2.

Ze srovnání počtu operací v tabulkách 2.1 a 2.2 vyplývá vyšší efektivnost Taylorovy řady. K dosažení kompletního výsledku x_1, y_1 a z_1 bylo nutné provést 18 výpočetních operací při použití metody Runge-Kutta, kdežto metoda Taylorovy řady potřebovala 14 výpočetních operací.

Pro potvrzení předešlých zajímavých výsledků bylo vypracováno srovnání pro metodu Runge-Kutta 4. řádu a metodu Talorovy řady 4. řádu. Detailně popsanou posloupnost výpočetních operací použitých při řešení pomocí obou metod zde nebudu uvádět kvůli její větší délce. Postup výpočtu je analogický s metodami 2. řádu - rovnice (2.31)–(2.48). Při použití metody Runge-Kutta 4.řádu každý mikroprocesor provádí 46 operací, u metody Taylorovy řady 4. řádu je to 28 operací.

Tabulka 2.2: Posloupnost operací při řešení soutavy metodou Taylorovy řady 2. řádu

Pořadí Procesor X	**Procesor Y**	**Procesor Z**	
1	$X1 = a_1 \cdot x_0$	$Y1 = a_2 \cdot x_0$	$Z1 = a_3 \cdot x_0$
2	$X2 = b_1 \cdot y_0$	$Y2 = b_2 \cdot y_0$	$Z2 = b_3 \cdot y_0$
3	$X3 = c_1 \cdot z_0$	$Y3 = c_2 \cdot z_0$	$Z3 = c_3 \cdot z_0$
4	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$
5	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$
6	$X6 = h \cdot X5$	$Y6 = h \cdot Y5$	$Z6 = h \cdot Z5$
7	$X7 = a_1 \cdot X6$	$Y7 = a_2 \cdot X6$	$Z7 = a_3 \cdot X6$
8	$X8 = b_1 \cdot Z6$	$Y8 = b_2 \cdot Y6$	$Z8 = b_3 \cdot Z6$
9	$X9 = c_1 \cdot Y6$	$Y9 = c_2 \cdot Y6$	$Z9 = c_3 \cdot Z6$
10	$X10 = X7 + X8$	$Y10 = Y7 + Y8$	$Z10 = Z7 + Z8$
11	$X11 = X10 + X9$	$Y11 = Y10 + Y9$	$Z11 = Z10 + Z9$
12	$X12 = h/2 \cdot X11$	$Y12 = h/2 \cdot Y11$	$Z12 = h/2 \cdot Z11$
13	$X13 = X12 + X6$	$Y13 = Y12 + Y6$	$Z13 = Z12 + Z6$
14	$X14 = x_0 + X13$	$Y14 = y_0 + Y13$	$Z14 = z_0 + Z13$

Z provedených rozborů je zřejmé, že při použití metody Runge-Kutta a Taylorovy řady stejných řádů, je počet výpočetních operací při výpočtu Taylorovou řadou menší. S použitím vyššího řádu metody je tento rozdíl ještě výraznější.

To například znamená, že při stejném počtu použitých operací jsme schopni metodou Taylorovy řady dosáhnout vyšší přesnosti (díky dosažení vyššího řádu metody) než metodou Runge-Kutta.

2.2.3 Metodika tvořících diferenciálních rovnic

Získané výsledky numerického řešení soustav homogenních lineárních diferenciálních rovnic s konstantními koeficienty byly použity jako podklady pro numerické řešení další skupiny obyčejných diferenciálních rovnic - pro soustavy nehomogenních lineárních diferenciálních rovnic s konstantními koeficienty.

Při řešení nehomogenní diferenciální rovnice se požadovaná funkce $f(t)$ ("pravá strana" diferenciální rovnice) "generuje" pomocí *tvořících diferenciálních rovnic* - v závislosti na typu funkce $f(t)$ se původní soustava diferenciálních rovnic rozšíří o další tvořící soustavu diferenciálních rovnic 1. řádu. Bližší seznámení s touto transformací je obsahem článku [13].

Např. rovnice (2.49)

$$y' = y + \sin(t) \qquad y(0) = y_0 \tag{2.49}$$

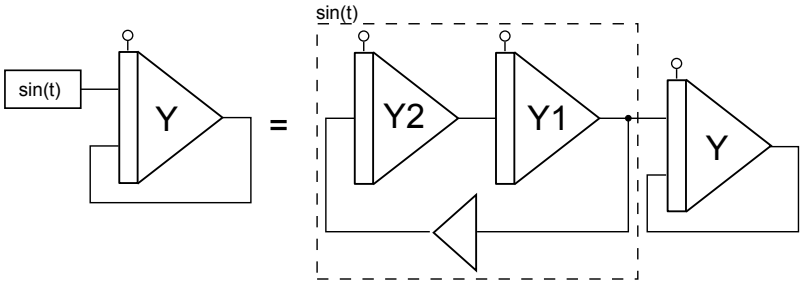
se převede na systém (2.50)–(2.52) obdobný soustavě rovnic (2.28)–(2.30)

$$y' = y + y_1 \qquad y(0) = y_0 \tag{2.50}$$

$$y_1' = y_2 \qquad y_1(0) = 0 \tag{2.51}$$

$$y_2' = -y_1 \qquad y_2(0) = 1 \tag{2.52}$$

Blokově se překreslí na schéma obsahující 3 integrátory (viz obrázek 2.3), které odpovídá nové soustavě diferenciálních rovnic (2.50)–(2.52).



Obrázek 2.3: Blokové schéma rovnice (2.49) před a po transformaci

Princip tvořících diferenciálních rovnic je charakteristický rovněž při řešení soustav diferenciálních rovnic s proměnnými koeficienty (tvořící diferenciální rovnice "vygenerují" požadované proměnné koeficienty).

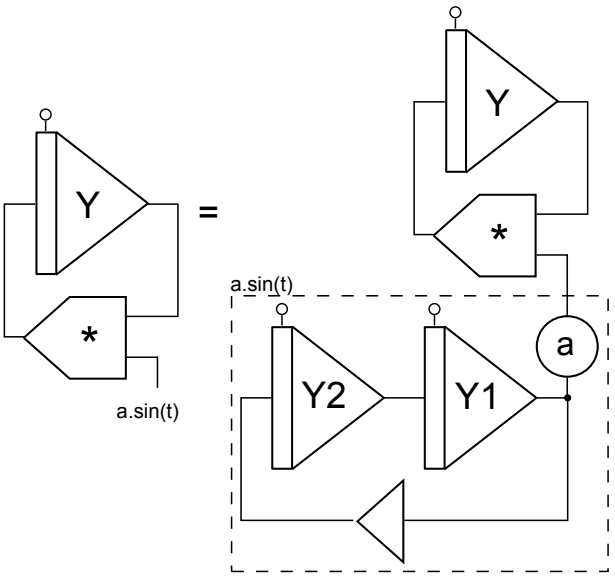
Např. při řešení rovnice (2.53)

$$y' = a{\cdot}y{\cdot}\sin(t) \qquad y(0) = y_0 \tag{2.53}$$

bude mít nově vzniklá soustava diferenciálních rovnic tvar

$$y' = a{\cdot}y{\cdot}y_1 \qquad y(0) = y_0 \tag{2.54}$$
$$y_1' = y_2 \qquad y_1(0) = 0 \tag{2.55}$$
$$y_2' = -y_1 \qquad y_2(0) = 1 \tag{2.56}$$

Na obrázku 2.4 je uvedeno blokové schéma odpovídající původní rovnici (2.53) a nově vzniklé soustavě diferenciálních rovnic(2.54)–(2.56). V obrázku je použit známý symbol násobičky.



Obrázek 2.4: Blokové schéma rovnice (2.53) před a po transformaci

Ke generování funkce $\sin(t)$, jak bylo vidět v předešlých dvou příkladech, je použita soustava dvou tvořících diferenciálních rovnic. Podobné výrazy (tvořící diferenciální rovnice) mohou být vytvořeny pro všechny elementární funkce, jako jsou $\exp, \sin, \cos, \tan, \coth, \ln, \sinh, \ldots$.

Potom může být zadaná diferenciální rovnice (pravá strana rovnice) rozložena na sekvenci jednoduchých operací: sčítání, odčítání, násobení, dělení a jejich kombinaci. Získání vyšších derivací těchto nově vzniklých rovnic (2.50)–(2.52) nebo (2.54)–(2.56) které se využívají v Taylorově řadě již není problém. Kompletní tabulka tvořících diferenciálních rovnic je obsažena v [Hol99].

Nově popsanou metodikou tvořících diferenciálních rovnic lze tedy řešit soustavy homogenních diferenciálních rovnic, nehomogenních diferenciálních rovnic, diferenciálních rovnic s proměnnými koeficienty i soustavy nelineárních diferenciálních rovnic.

Pro metodiku tvořících diferenciálních rovnic je charakteristické, že dokonce již jednu nelineární diferenciální rovnici lze řešit pomocí paralelně spolupracujících procesorů (integrátorů). Obecně je počet paralelně spolupracujících procesorů (interátorů) roven počtu diferenciálních rovnic 1. řádu, na který byl zadaný problém převeden.

2.3 Shrnutí

Jak bylo vidět v této kapitole, každou diferenciální rovnici, resp. soustavu rovnic, lze převést na ekvivalentní blokové schéma. Tento způsob je typický pro analogové počítače, kde se podle blokového (analogového) schématu provedlo zapojení jednotlivých prvků a následně jsme obdrželi požadované řešení. Bližší popis analogových počítačů je v [BHSL82] a [Mac07]. Abychom toto analogové schéma, resp. jemu odpovídající soustavu diferenciálních rovnic, vyčíslili v diskrétním výpočetním systému, je potřeba realizovat všechny elementární prvky v podobě diskrétních výpočetních jednotek.

Metoda Taylorovy řady poskytuje velice přesné řešení obyčejných diferenciálních rovnic. Ze srovnání obsaženého v této kapitole je zřejmé, že při použití metody Runge-Kutta a Taylorovy řady stejných řádů, je počet výpočetních operací při použití Taylorovy řady menší. Tento rozdíl se ještě zvětší, pokud se použije vyšší řád metody.

Kapitola 3

Návrh paralelního systému

Z předchozího rozboru, který je pro názornost proveden pro homegenní soustavu diferenciálních rovnic, vyplývá, že v závisloti na typu řešených úloh musí být při návrhu paralelního systému kladen důraz na výběr vhodné architektury procesoru (aritmeticko-logické jednotky). Dále musí být použit vhodný typ propojovací sítě, případně kombinace více různých sítí. Největší problémy nastávají při návrhu velmi rozsáhlých paralelních systémů, kde musí být efektivně propojeny stovky až tisíce výpočetních uzlů. V neposlední řadě je také důležité řízení celého výpočetního a propojovacího systému.

Použité výpočetní bloky a jejich propojení vychází ze schémat, která byla použita v kapitole 2. Tzn. výpočetní jednotky provádí operace, které odpovídají integrátorům, sčítačkám případně odčítačkám, násobičkám a děličkám. Není problém navrhnout sčítačku případně násobičku. Otázkou ale je návrh integrátoru. Navržené výpočetní, propojovací a řídící podsystémy jsou obsahem této kapitoly. Na závěr je popis propojení těchto podsystémů do jednoho celku - paralelního systému.

3.1 Koncepce aritmeticko-logické jednotky

Základ celého systému tvoří aritmeticko-logické jednotky (ALU). ALU provádí vlastní výpočet numerické integrace. Aritmeticko-logická jednotka tedy představuje integrátor. Dá se očekávat, že stovky až tisíce aritmeticko-logických jednotek mohou být propojeny a může být řešena velmi rozsáhlá soustava diferenciálních rovnic. ALU je proto koncipována jako specializovaná jednoúčelová jednotka, provádějící základní matematické operace sčítání a násobení.

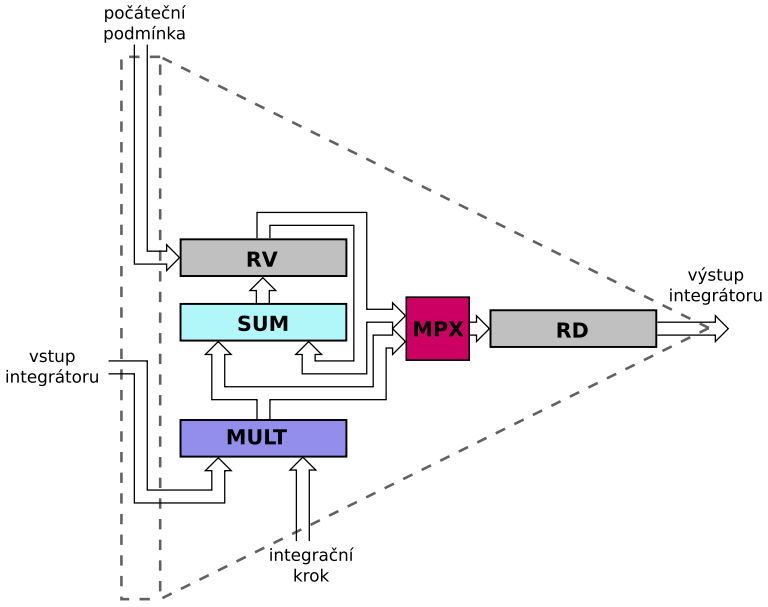
Jak je popsáno v kapitole 2, nebo přímo v části 2.2 popisující metodu Taylorovy řady, zadaný problém je převeden na soustavu homogenních lineárních diferenciálních rovnic s konstantními koeficienty. Při numerickém výpočtu těchto diferenciálních rovnic Taylorovou řadou se používají dvě základní operace: **násobení** (výpočet DYp_i) a **sčítání** (iterativní výpočet y_{i+1}).

Tyto dvě základní operace mohou být prováděny sériově nebo paralelně. Podobně komunikace mezi procesory se může řešit sériově nebo paralelně. Podle toho budeme v následujícím textu rozdělovat integrátory do těchto skupin:

- paralelně-paralelní integrátory (paralelní komunikace a paralelní výpočet)
- sériově-paralelní integrátory (sériová komunikace a paralelní výpočet)
- sériově-sériové integrátory (sériová komunikace a sériový výpočet)

3.1.1 Paralelně-paralelní integrátor

Násobení je realizováno paralelní násobičkou a sčítání paralelní sčítačkou. Blokové schéma je zobrazeno na obrázku 3.1. Čárkovaně je vyznačen symbol integrátoru.



Obrázek 3.1: Blokové schéma paralelního integrátoru

Význam jednotlivých bloků:

RV	registr výsledku
RD	registr součinu
MPX	multiplexor
SUM	paralelní sčítačka
MULT	paralelní násobička

Funkce integrátoru: cyklus je zahájen tím, že se do registrů RD a RV uloží hodnota y_i. Na vstupu se objeví hodnota $f(y_i)$, což je výstupní hodnota prvku, který je připojen na vstup integrátoru. Integrační krok se nataví na velikost h. Součin (z násobičky MULT) těchto dvou vstupů se přepíše do registru RD a současně se přičte k registru RV. V RD je tedy hodnota $DY1$ a v RV je mezisoučet $y_i + DY1$. V dalším kroku se na vstupu objeví $f(DY1)$ a integrační krok $h/2$. Jejich roznásobením se vypočítá $DY2$, jež se opět uloží do RD sečte s RV. Celý cyklus se opakuje do té doby, dokud není dosaženo požadované přesnosti, nebo maximálního počtu iterací, čímž získáme y_{i+1}.

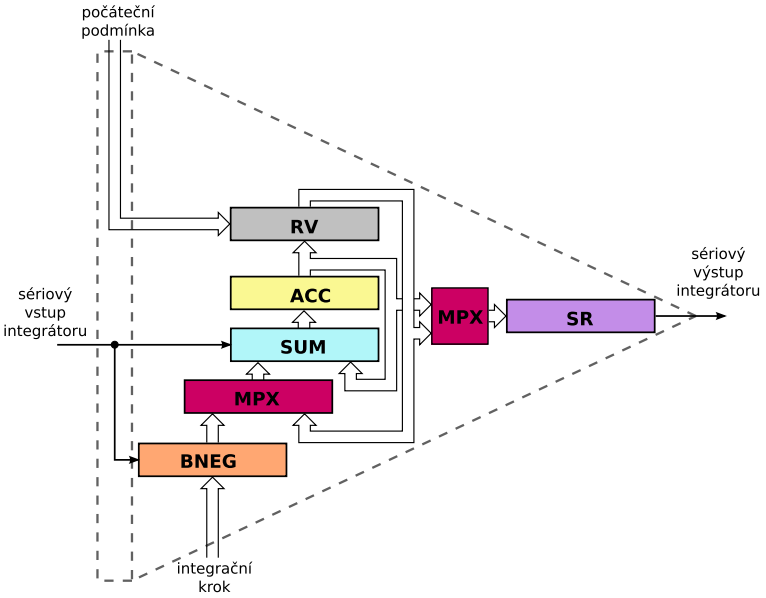
Tento typ integrátoru je nejrychlejší, čas výpočtu jednoho členu Taylorovy řady je:

$$t_{PP} = \tau_{nas} + \tau_{sec} + \tau_{sit} \tag{3.1}$$

tedy je dán součtem času násobení τ_{nas} a sčítání τ_{sec}, případně i zpožděním signálů v propojovací síti τ_{sit}. Ovšem za cenu největší složitosti zapojení, na kterém má největší podíl kombinační násobička. Dalším nepříznivým kritériem je počet vstupů a výstupů, který je přímo úměrný šířce paralelní datové sběrnice.

3.1.2 Sériově-paralelní integrátor

V této variantě se násobení provádí sekvenční metodou na principu Boothova algoritmu násobení. Sčítání se však nadále provádí paralelně v jednom kroku.



Obrázek 3.2: Blokové schéma sériově-paralelního integrátoru

Význam jednotlivých bloků:

RV	registr výsledku
MPX	multiplexor
SUM	paralelní sčítačka
ACC	akumulátor
SR	posuvný registr
BNEG	obvod řízené negace (pro sekvenční násobení)

Princip výpočtu sériově-paralelního integrátoru je následující: V registru RV a SR je uloženo y_i. Vstup integračního kroku má hodnotu h, MPX je přepnut na cestu od bloku řízené negace BNEG. Výpočet y_{i+1} pak probíhá takto: nejprve se akumulátor ACC vynuluje, na vstupu je připraven nejméně významový bit $f(y_i)$. Tento bit určí, zda se bude násobenec z registru RN přičítat do akumulátoru ACC, odečítat (vytvoří se záporná podoba násobence - druhý doplněk), nebo zda se bude ignorovat (vynuluje). Výsledek ze sčítačky se zapíše do ACC a celý akumulátor a posuvný registr SR se potom posune o jedno místo doprava.

Tento postup se opakuje, dokud není přijat poslední bit z $f(y_i)$ a následně zpracován. Tímto se dosáhne vynásobení h a $f(y_i)$. Tento výsledek násobení se uloží do posuvného registru SR. Multiplexor se přepne místo z BNEG na RV a výsledek násobení DYp (uložený v ACC) se sečte s hodnotou uloženou v registru výsledku RV a uloží se do ACC a následně do RV. Celý cyklu se opakuje do té doby, dokud není dosaženo požadované přesnosti, nebo maximálního počtu iterací, čímž získáme y_{i+1}.

Výhodou tohoto přístupu je malý počet potřebných vývodů rozhraní zapojení, protože data vstupují i vystupují sériově. Chceme-li zpřesnit výpočet zvětšením počtu bitů, na nichž jsou čísla zobrazena, pak se, na rozdíl od předchozí varianty, v celkovém zapojení nic nezmění, "pouze" se změní šířka registrů, sčítačky a prodlouží se posloupnost řídících

signálů, ale rozhraní integrátoru, stejně jako propojovací síť, zůstanou zachovány.

Nevýhoda oproti předchozí variantě je zpomalení, protože násobení je zde prováděno v n krocích (n = počet bitů zobrazení čísel). Čas výpočtu jednoho členu Taylorovy řady je:

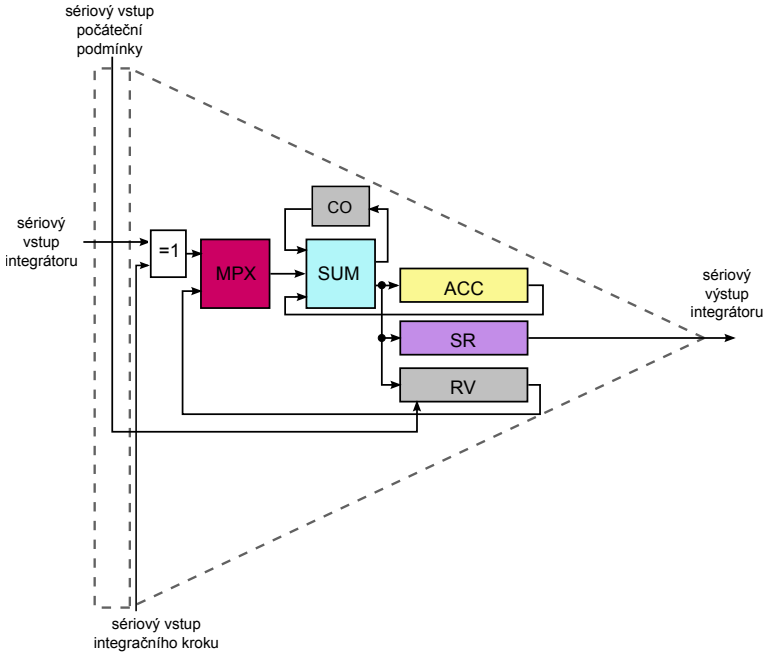
$$\begin{aligned} t_{SP} &= \tau_{nas} + \tau_{sec} + \tau_{sit} \\ t_{SP} &= n{\cdot}\tau_{sec} + \tau_{sec} + \tau_{sit} \end{aligned} \tag{3.2}$$

tedy je dán součtem času násobení (což je vlastně n kroků sčítání $n{\cdot}\tau_{sec}$) a nezbytné přičtení τ_{sec} výsledku násobení k předešlému celkovému výsledku, případně i zpožděním signálů v propojovací síti τ_{sit}.

Jistá možnost optimalizace této varianty integrátoru je v použití překódování více bitů najednou (Boothovo překódování s radixem 4 či 8). K mezivýsledku násobení se pak přičítá násobek integračního kroku, který je určen překódováním vstupní skupiny bitů. Tímto je možno počet potřebných kroků pro násobení zmenšit. Při této variantě nám už ale nestačí pouze vytvářet kladnou a zápornou hodnotu násobence. Musí být použit obvod, který bude vytvářet překódování 2 případně 4 bity najednou. To nám celé zapojení zesložití.

3.1.3 Sériově-sériový integrátor

Tato varianta vychází z principu sériově-paralelního integrátoru. Na rozdíl od předchozí varianty však provádí sekvenčně nejen násobení, ale i operaci sčítání.



Obrázek 3.3: Blokové schéma sériového integrátoru

Význam jednotlivých bloků:

SUM	úplná jednobitová sčítačka
CO	klopný obvod pro uchování přenosu
MPX	multiplexor
ACC	akumulátor
RV	posuvný registr výsledku
SR	výstupní posuvný registr

Funkce je tato: Nejprve se vynuluje ACC, do RV se vloží hodnota y_i. Obvod uchování přenosu CO se vynuluje. Multiplexor MPX se nastaví na cestu z ACC. Na vstupu integrátoru se objeví nejméně významový bit $f(y_i)$ a na sériovém vstupu integračního kroku se postupně objevují jednotlivé bity integračního kroku, počínaje nejméně významovým bitem. Tato posloupnost se v závislosti na hodnotě vstupu integrátoru sériově přičte k akumulátoru. Po dokončení výpočtu mezivýsledku se na vstupu objeví významnější bit $f(y_i)$ a současně se posune výstupní registr SR. Celý postup se opakuje do té doby, až se na vstupu integrátoru objeví poslední (nejvýznamnější) bit $f(y_i)$. Po dokončení operace násobení se hodnota uložená v ACC (DYp) uloží do výstupního registru SR a sériově se přičte k hodnotě uchované v registru výsledku.

Tato varianta má opět menší nároky na zapojení, ale cena, kterou je v tomto případě počet cyklů, jež jsou potřeba na celý výpočet (a tedy i čas), je dána exponenciálním vztahem.

$$\begin{aligned} t_{SS} &= \tau_{nas} + \tau_{sec} + \tau_{sit} \\ t_{SS} &= n{\cdot}n{\cdot}\tau_{sec} + n{\cdot}\tau_{sec} + \tau_{sit} \\ t_{SS} &= n^2{\cdot}\tau_{sec} + n{\cdot}\tau_{sec} + \tau_{sit} \end{aligned} \tag{3.3}$$

τ_{sec} v této verzi představuje čas sčítání úplné jednobitové sčítačky, n je počet bitů, na nichž jsou uloženy hodnoty násobitele i násobence.

3.2 Rozšíření integrátorů

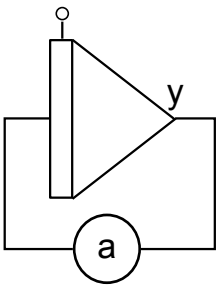
Samotné integrátory, představené výše, slouží k řešení homogenních diferenciálních rovnic typu (2.13). Pro řešení všech typů rovnic vzniklých po provedení transformace pomocí tvořících diferenciálních rovnic je nutné realizovat další operace a to operaci násobení konstantou, při řešení soustav operaci sčítání jednotlivých diferenciálních rovnic (viz např. soustava rovnic a její řešení (2.40)–(2.48)) a násobení jednotlivých diferenciálních rovnic mezi sebou (viz např. rovnice (2.54)).

3.2.1 Násobení konstantou

Při řešení diferenciální rovnice s konstantními koeficienty:

$$y' = a{\cdot}y \qquad y(0) = y_0 \tag{3.4}$$

Zapojení tohoto typu úlohy je na obrázku 3.4.



Obrázek 3.4: Blokové schéma násobení konstantou

Zápis Taylorovy řady lze obdržet ve známém tvaru:

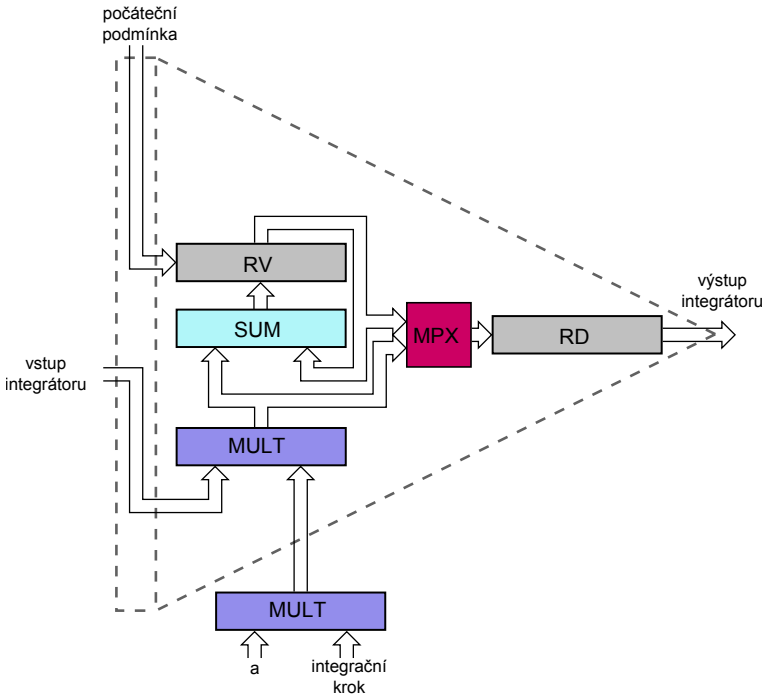
$$y_{(i+1)} = DY0_i + DY1_i + DY2_i + DY3_i + \cdots \tag{3.5}$$

kde význam jednotlivých členů je:

$$\begin{aligned} DY0_i &= a{\cdot}y_i \\ DY1_i &= a{\cdot}h{\cdot}DY0_i \\ DY2_i &= a\frac{h}{2}\,DY1_i \\ DY3_i &= a\frac{h}{3}\,DY2_i \end{aligned}$$

Možné řešení je, že se do přípravných výpočtů zahrne nejen h/p, ale také $a{\cdot}h/p$. Tato možnost bude pro probíhající výpočet v paralelním sytému nejrychlejší a nezabere žádné místo na čipu navíc. Na druhé straně výpočet a distribuce jednotlivých podílů integračního kroku h/p resp. $a{\cdot}h/p$ bude složitější. Už nebude použita v každém integrátoru stejná hodnota, ale musíme zajistit správné nahrání podle řešených diferenciálních rovnic.

Nabízí se i kombinační varianta (relativně velmi nákladná) viz obrázek 3.5, kde je zobrazeno řešení pro paralelně-paralelní verzi integrátoru.



Obrázek 3.5: Technická realizace přednásobení konstantou

Úprava spočívá v tom, že se na vstup pro integrační krok připojí násobička. Na vstupy této násobičky se přivede aktuální podíl integračního kroku h/p a požadovaná konstanta a. Tyto dvě hodnoty se nezávisle na výpočtu v integrátoru vynásobí a výsledek $a{\cdot}h/p$ je připraven na vstupu integrátoru určený pro integrační krok. I v této variantě je důležité před začátkem výpočtu zajistit distribuci konstantních koeficientů ke správným integrátorům resp. násobičkám připojených k těmto integrátorům.

Můžeme také použít k řešení tohoto problému navržených integrátorů. Ty umí operaci násobení provádět. Museli bychom však vyrobit tento integrátor univerzálnější, aby umožňoval při výpočtu jednoho členu Taylorovy řady provádět dvě násobení - $h{\cdot}a$ a následně tento výsledek vynásobit s $DY(p-1)_i$, čímž vypočítáme DYp_i.

Chtěli bychom, aby každý integrátor pracoval stejně, tzn. byl řízen z jedné společné řídící jednotky. Kdyby měl vykonávat tato dvě násobení, museli bychom na vstup pro integrační krok přivádět integrační krok a zadaný koeficient a vše řídit řídící jednotkou. Všechny integrátory by zřejmě nemusely provádět obě násobení - tzn. některé by čekaly, až zase budou moci pokračovat ve společném výpočtu (= náročnější synchronizace).

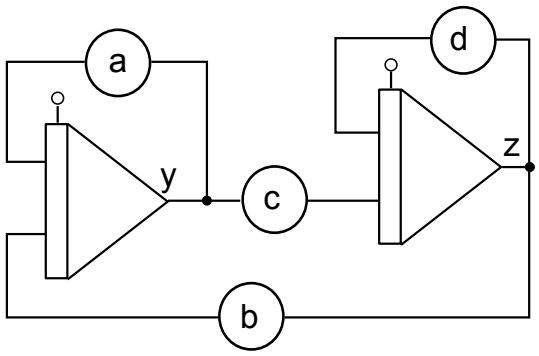
Jako lepší řešení se jeví přidat do systému násobičky, které by součin $a{\cdot}h$ samostatně vynásobily a potom ho přivedly na vstup pro integrační krok do integrátoru. Integrátory by čekaly pouze při prvním výpočtu součinu, každý další výpočet součinu $a{\cdot}h/p$ by už probíhal současně s výpočtem integrátoru a byl připraven na vstupu integrátoru dříve, než ho bude potřebovat. To nám přináší do našeho systému možnost zřetězeného zpracování - pipeline.

3.2.2 Součtový integrátor

Problém vícevstupého součtového integrátoru se objeví při řešení soustavy diferenciálních rovnic typu:

$$y' = a{\cdot}y + b{\cdot}z \qquad y(0) = y_0 \tag{3.6}$$
$$z' = c{\cdot}y + d{\cdot}z \qquad z(0) = z_0 \tag{3.7}$$

Obecné zapojení tohoto typu úloh je na obrázku 3.6.



Obrázek 3.6: Zapojení s vícevstupým součtovým integrátorem

Pokud na zadanou diferenciální rovnici aplikujeme metodu Taylorovy řady, dostáváme řešení:

$$y_{(i+1)} = y_0 + DY1_i + DY2_i + DY3_i + \cdots$$
$$z_{(i+1)} = y_0 + DZ1_i + DZ2_i + DZ3_i + \cdots$$

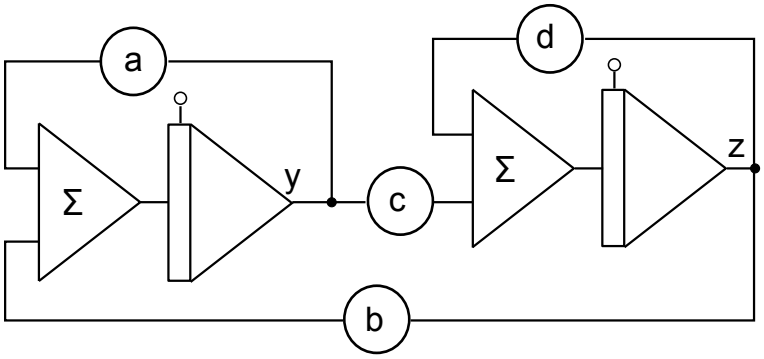
kde význam jednotlivých členů je:

$$DY1_i = h(a{\cdot}DY0_i + b{\cdot}DZ0_i) \qquad DZ1_i = h(c{\cdot}DY0_i + d{\cdot}DZ0_i)$$
$$DY2_i = \tfrac{h}{2}(a{\cdot}DY1_i + b{\cdot}DZ1_i) \qquad DZ2_i = \frac{h}{2}(c{\cdot}DY1_i + d{\cdot}DZ1_i)$$
$$DY3_i = \tfrac{h}{3}(a{\cdot}DY2_i + b{\cdot}DZ2_i) \qquad DZ3_i = \frac{h}{3}(c{\cdot}DY2_i + d{\cdot}DZ2_i)$$

Existuje více možností realizace tohoto řešení, které si dále představíme.

Použití bloku vícevstupá sčítačka

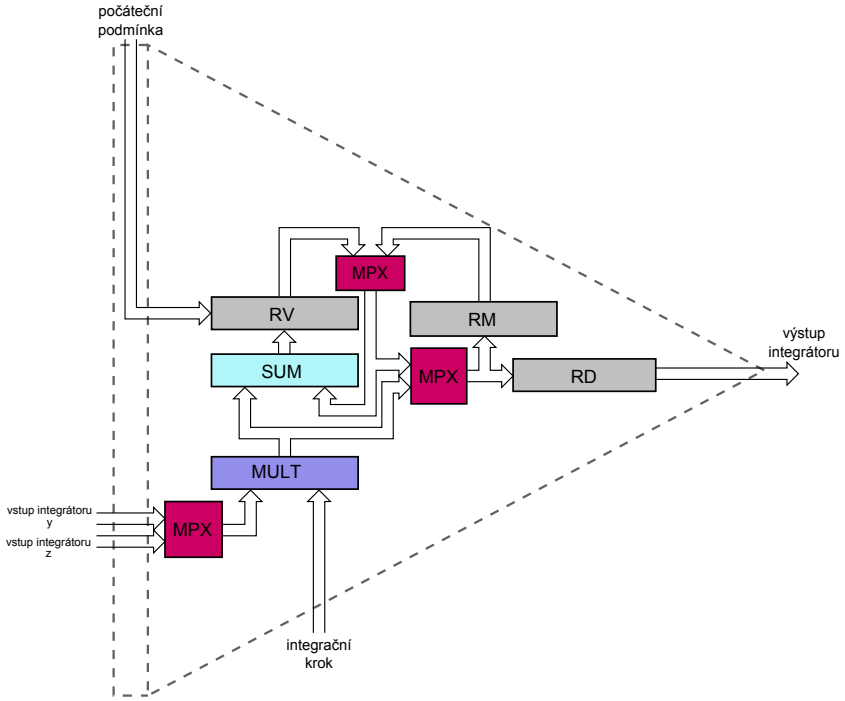
Základní a nejjednodušší možností je zařadit do systému víceoperandovou sčítačku. Při této variantě řešení není potřeba nijak upravovat navržené integrátory. Jen se do systému přidají sčítačky, které budou zapojeny jako na obrázku 3.7. Počet operandů sčítačky určuje i počet operandů obsažených v diferenciální rovnici (v našem příkladu dva).



Obrázek 3.7: Součtový integrátor - využití sčítačky a jednovstupého integrátoru

Přepínání vstupů integrátoru (sekvenční integrátor)

Blokové schéma integrátoru, který pracuje na tomto principu je uveden na obrázku 3.8. Nejdříve se provede výpočet odpovídající prvnímu vstupu integrátoru (např. vstup y), potom se dopočítá výpočet odpovídající druhému (vstup z).



Obrázek 3.8: Přepínání vstupů integrátoru

V této variantě použijeme výše navržený jednovstupý integrátor tak, že napřed provedeme

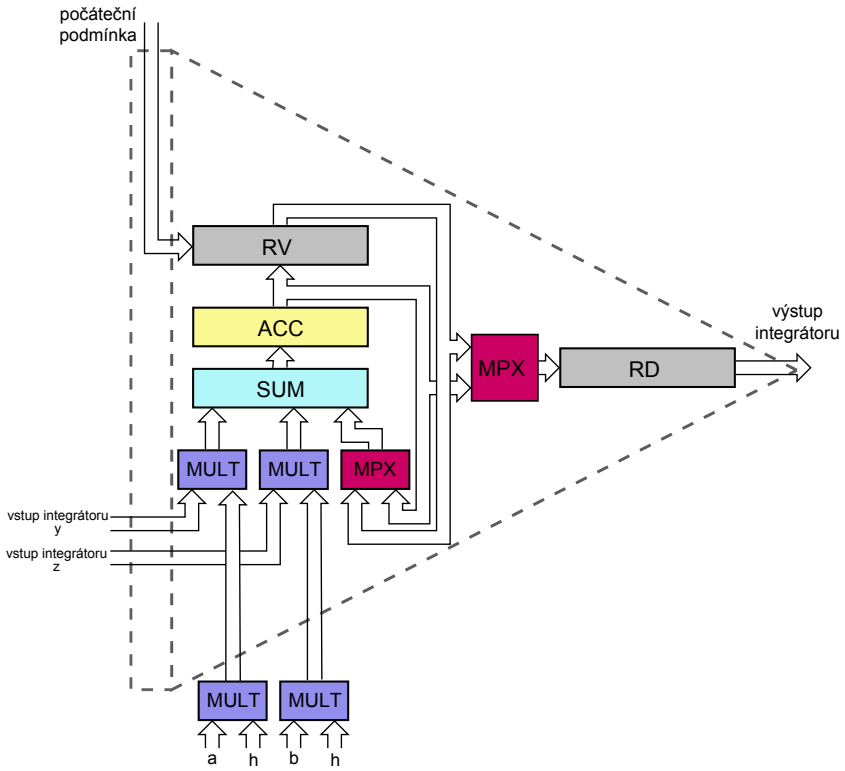
jedno vynásobení $(a{\cdot}h{\cdot}DY(p-1)_i)$, které si musíme v integrátoru uložit. Potom přepneme integrátor na druhý vstup a provedeme druhé vynásobení $(b{\cdot}h{\cdot}DZ(p-1)_i)$. Nakonec provedeme sečtení těchto dvou hodnot a získáme výsledek (DYp_i). Tzn. že musíme navržené integrátory rozšířit o registr RM, který budou sloužit k uchování vypočtených hodnot.

Vnitřní zapojení není závislé na počtu operandů obsažených v diferenciální rovnici. Jen se musí odpovídajícím způsobem rozšířit počet vstupů multiplexoru, který přepíná vstupy integrátoru. Jedná se v podstatě o sériový výpočet, kdy jsou postupně zpracovávány jednotlivé vstupy. Z toho vyplývá delší doba výpočtu než v předchozí variantě.

Řízení a synchronizace celého systému bude složitějším a ztratí se univerzálnost navrženého integrátoru. Nastala by situace, že některé integrátory (pouze s jedním vstupem) budou čekat na ty, které počítají s více vstupy (čekají na přepnutí dalšího vstupu).

Vícevstupý kombinační integrátor

Další variantou řešení soustavy rovnic (3.6), (3.7) je návrh vícevstupého integrátoru, který bude oba vstupy zpracovávat současně. Úprava spočívá v použití vícevstupé sčítačky. Princip tohoto rozšíření je demonstrován na obrázku 3.9.



Obrázek 3.9: Vícevstupý integrátor

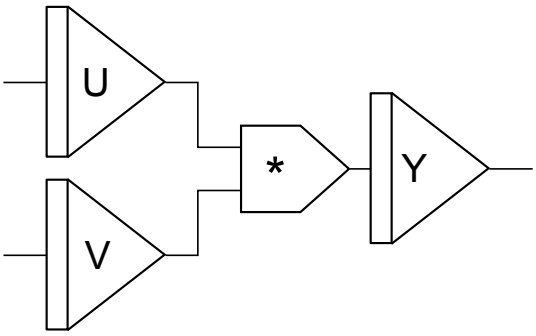
Obvod je samozřejmě nákladný, ale nejrychlejší (vhodný pro jednoúčelové real-time výpočty). Opět toto řešení není zcela univerzální, protože takové integrátory musí mít odlišné řízení od jednovstupého integrátoru. Čas strávený čekáním jednovstupého integrátoru na tento vícevstupý by už ale nebyl tak velký jako u varianty s přepínáním vstupů. Větší počet operandů neprodlužuje zásadně dobu výpočtu jako předchozí varianta, ale celé zapojení se stává podstatně složitější.

3.2.3 Násobící integrátor

Následující integrátor řeší diferenciální rovnici typu:

$$y' = u \cdot v \qquad y(0) = y_0 \tag{3.8}$$

Obecné zapojení tohoto typu úloh je na obrázku 3.10.



Obrázek 3.10: Princip realizace násobícího integrátoru

Zápis Taylorovy řady lze opět obdržet ve známém tvaru:

$$y_{(i+1)} = y_i + DY1_i + DY2_i + DY3_i + DY4_i + \cdots \tag{3.9}$$

Nyní ale výpočet jednotlivých členů bude poněkud obtížnější, protože jde o derivaci součinu. Výpočet jednotlivých derivací je zobrazen dále:

$$\begin{aligned}
y' &= u \cdot v && (3.10)\\
y'' &= u' \cdot v + u \cdot v' && (3.11)\\
y''' &= u'' \cdot v + u' \cdot v' + u' \cdot v' + u \cdot v'' = u'' \cdot v + 2u' \cdot v' + u \cdot v'' && (3.12)\\
y^{IV} &= u''' \cdot v + u'' \cdot v' + 2u'' \cdot v' + 2u' \cdot v'' + u' \cdot v'' + u \cdot v''' = \\
&= u''' \cdot v + 3u'' \cdot v' + 3u' \cdot v'' + u \cdot v''' && (3.13)\\
&\vdots
\end{aligned}$$

Po použití již známého označení bude tvar jednotlivých derivací:

$$\begin{aligned}
DY1 = h \cdot y' \Rightarrow y' = \tfrac{DY1}{h} \qquad & u' = \tfrac{DU1}{h} \qquad & v' = \tfrac{DV1}{h}\\
DY2 = \tfrac{h^2}{2!} y'' \Rightarrow y'' = \tfrac{DY2}{\frac{h^2}{2!}} \qquad & u'' = \tfrac{DU2}{\frac{h^2}{2!}} \qquad & v'' = \tfrac{DV2}{\frac{h^2}{2!}}\\
DY3 = \tfrac{h^3}{3!} y''' \Rightarrow y''' = \tfrac{DY3}{\frac{h^3}{3!}} \qquad & u''' = \tfrac{DU3}{\frac{h^3}{3!}} \qquad & v''' = \tfrac{DV3}{\frac{h^3}{3!}}\\
DY4 = \tfrac{h^4}{4!} y^{IV} \Rightarrow y^{IV} = \tfrac{DY4}{\frac{h^4}{4!}} \qquad & u^{IV} = \tfrac{DU4}{\frac{h^4}{4!}} \qquad & v^{IV} = \tfrac{DV4}{\frac{h^4}{4!}}
\end{aligned}$$

Nyní se vyjádřené derivace z předešlého kroku dosadí do rovnic (3.10)–(3.13) a vyjádří se členy DYp. Následně již obdržíme tvar jednotlivých členů v rovnici (3.9):

$$DY1_i = h{\cdot}(u_i{\cdot}v_i) \tag{3.14}$$

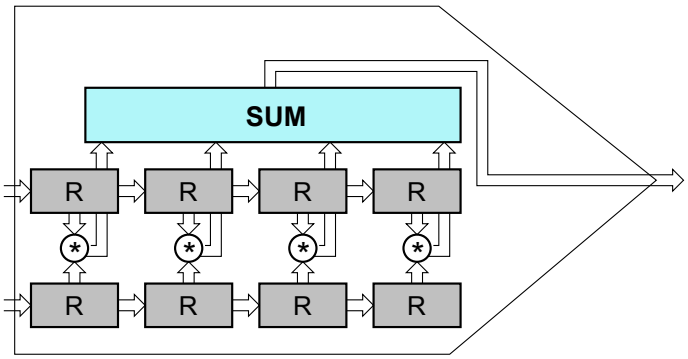
$$DY2_i = \frac{h}{2}(DU1_i{\cdot}v_i + u_i{\cdot}DV1_i) \tag{3.15}$$

$$DY3_i = \frac{h}{3}(DU2_i{\cdot}v_i + DU1_i{\cdot}DV1_i + u_i{\cdot}DV2_i) \tag{3.16}$$

$$DY4_i = \frac{h}{4}(DU3_i{\cdot}v_i + DU2_i{\cdot}DV1_i + DU1_i{\cdot}DV2_i + u_i{\cdot}DV3_i) \tag{3.17}$$

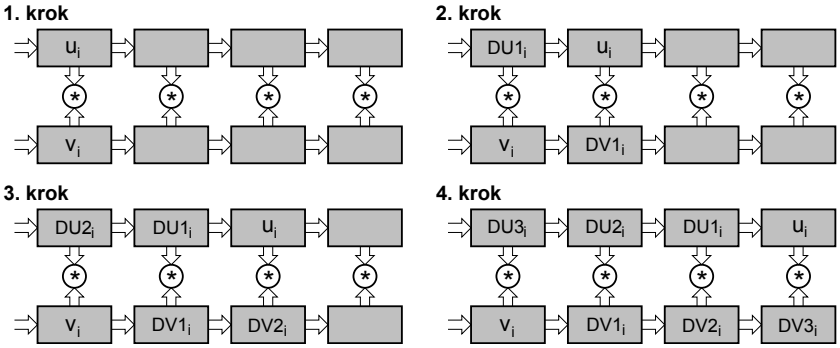
Jak je vidět z obsahu členů DYp_i, v jednotlivých výpočetních krocích (3.14)–(3.17)potřebujeme mít k dispozici všechny předešlé hodnoty DUp_i a DVp_i. Technická realizace "pseudo násobičky" předřazené integrátoru Y na obrázku 3.10 s tím proto musí počítat.

Celý výpočet DYp_i se dělí do dvou kroků. Násobička provádí operaci násobení a následné sečtení výsledků každého násobení - členy v závorce v rovnicích (3.14)–(3.17). Tento výpočet je velmi blízký principu, který využívá operace konvoluce a označuje se multiply-accumulate. Princip této násobičky je zobrazen na obrázku 3.11. Vzniklé číslo přivedeme na vstup integrátoru Y, který je klasické koncepce a provede vynásobení integračním krokem h/p.



Obrázek 3.11: Princip realizace násobičky

Na příkladu jsou k uložení jednotlivých hodnot použity registry. Při tomto způsobu realizace se musí zajistit posouvání hodnot ze vstupu do registrů tak, aby byl výpočet správný. Postup korektního šíření je zobrazen na obrázku 3.12.



Obrázek 3.12: Ukázka šíření operandů registry násobičky

Možností technického řešení je samozřejmě více. Může být například použita paměť.

Ideální by byla paměť s více výstupními branami, které budou připojeny na jednotlivé násobičky. O správné adresování by se musel starat přídavný řídící obvod.

Celková koncepce násobičky je velmi závislá na technologii, která je použita při realizaci paralelního systému. Je zřejmé, že tato komponenta bude mít největší vliv na rychlost výpočtu celého paralelního systému. Všechny integrátory musí čekat na dokončení výpočtu v násobičce. Jak je vidět v rovnicích (3.14)–(3.17), s použitím vyššího řádu metody roste i počet potřebných operací násobení a přičtení. Pokud bude systém provozován v některých kritických situacích, bude rychlejší, když se vyhneme použití vyššího řádu metody Taylorovy řady. Toho můžeme dosáhnout např. použitím menšího integračního kroku.

3.3 Řadič - generátor řídících signálů

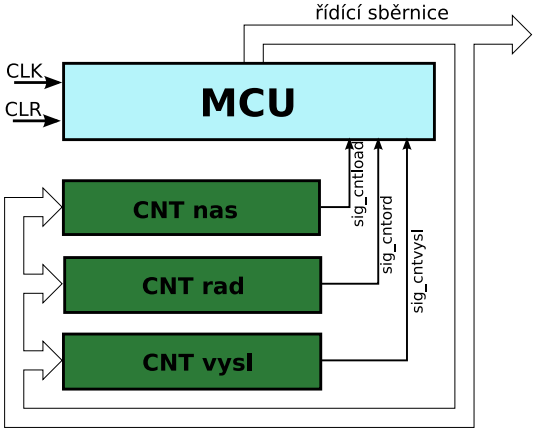
Funkce řadiče spočívá v generování posloupnosti řídících signálů pro procesor (integrátor) tak, aby plnil požadovanou funkci.

Existují dva základní přístupy v návrhu řadičů:

- **obvodový** - základem je dekodér instrukce, který podle typu instrukce, stavových bitů procesoru a aktuálního stavu vygeneruje odpovídající řídící signály pro obvod.
- **mikroprogramový** - základem je paměť mikroprogramu, ze které se podle přiložené adresy vybere požadovaná instrukce - operace, která už přímo generuje řídící signály pro obvod. Každá instrukce obsahuje adresu následující instrukce, která se má provádět.

Podrobný popis řadičů nalezneme v [Drá95] a [Hwa06, Chapter 10: Control Units]. Detailní popis návrhu mikroprogramového řadiče je v [PP06, Kapitola 14: Mikroprogramový automat].

Specializovaný řadič pro náš systém se příliš nepodobá univerzálním řadičům, protože je zde pevně stanoven sled řídících signálů, a není proto potřebný žádný dekodér instrukcí. V našem případě se dá použít řadič vycházející z modifikace mikroprogramového řadiče.



Obrázek 3.13: Generátor řídících signálů

Důležitou částí námi navrhovaného řadiče je počítání smyček. Hlavní smyčka počítá počet výsledků (y_i), další smyčka počítá dosažený řád Taylorovy řady (DYp_i). Další smyčku tvoří počítání délky slova (dílčích součtů při násobení) u sériových variant integrátorů a případně počet součtů úplné jednobitové sčítačky u sério-sériového integrátoru.

Pro realizaci smyček se přímo nabízí použití čítačů, které by přímo ovládal řadič. Na obrázku 3.13 je uvedeno blokové schéma specializovaného řadiče využívajícího čítače.

Význam jednotlivých čítačů je následující:

- **CNT nas** - počítá délku slova (počet posuvů SR a ACC) - dílčích součtů při násobení (u sériově-paralelní varianty integrátoru) případně počet součtů sčítačky (u sério-sériového integrátoru)
- **CNT rad** - počítá řád Taylorovy řady (DYp_i). Lze spojit s komparátorem, který bude porovnávat poslední dva dosažené výsledky. Pokud budou stejné, nemusíme již zbytečně počítat dále (nedosáhneme již žádného zpřesnění)
- **CNT vysl** - počítá požadovaný počet výsledků (y_i)

Další částí, o kterou může být řadič rozšířen, je komparátor. Ten plní funkci "omezovače" řádu Taylorovy řady. To znamená, že pokud se aktuální výstupní hodnota sledovaného integrátoru shoduje s jeho předchozí hodnotou, generuje komparátor signál, který může přerušit výpočet následujícího řádu. Tím se zamezí výpočtu, který už nezpřesňuje výpočet, ale přispívá pouze k jeho zpomalení.

3.4 Propojovací síť

3.4.1 Propojení jednotlivých aritmeticko-logických jednotek

Jedním z úkolů propojovacího systému je realizace propojení vstupů a výstupů jednotlivých mikroprocesorů mezi sebou.

Pro náš systém je jako propojovací síť nejvhodnější obyčejné statické propojení mezi prvky, podle konkrétního výpočetního schématu (řešené soustavy diferenciálních rovnic). Budeme-li však chtít, aby systém byl schopen provést výpočet obecné diferenciální rovnice, musíme použít univerzální statickou či dynamickou propojovací síť. Použití statického propojení podle řešeného zadání je možné pouze v případě, že budeme pro implementaci používat rekonfigurovatelnou technologii. Zde nemusíme mít pevně danou topologii sítě, ale vlastní statické propojení se nastaví při inicializaci celého paralelního systému. Tento přístup je nejrychlejší a nenáročný na další řízení.

Chceme-li použít takovou síť, na níž by bylo možné použít jakékoliv výpočetní schéma, nemůžeme uvažovat o použití nějaké obecné statické topologie uvedené v [DT04] nebo [Dvo04]. Hlavní důvod je ten, že nemůžeme zaručit, že libovolné dva integrátory spolu budou moci komunikovat. To můžeme zaručit pouze u úplného propojení, kde je každý prvek propojen se všemi ostatními. Tato verze má však značné omezení týkající se počtu propojených uzlů. Je použitelná jen pro malé systémy. Navíc by každý integrátor musel na vstupu obsahovat logiku, která by zajistila maskování nepotřebných vstupních signálů.

Při použití dynamické (nepřímé) propojovací sítě se provede před spuštěním vlastního výpočtu inicializace sítě (nastavení cest ve směrovacích prvcích). Dále se už propojení sítě nebude měnit. Propojovací cesta tvoří významnou složku zpoždění. Z tohoto důvodu je nevhodné použití víceúrovňových zapojení.

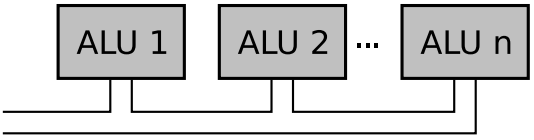
Nejvhodnější propojovací síť (pro nepříliš rozsáhlé systémy) je křížový přepínač. Má ze všech dynamických sítí nejmenší zpoždění, je jednoúrovňový a umožňuje propojení jednoho výstupu na více vstupů (větvení v blokovém schématu).

3.4.2 Datové sběrnice pro nahrání počátečních dat a výsledku

Propojovací systém dále slouží k napojení procesorů na řídící obvody, ze kterých se nahrávají hodnoty integračního kroku h a počáteční podmínky y_0 do jednotlivých aritmeticko-logických jednotek a po výpočtu slouží k obdržení hodnoty výsledku.

Jak bylo uvedeno výše, můžeme systémy z hlediska jejich komunikace rozdělit na sériové a paralelní. Z hlediska principiálního však mezi nimi není nijak významný rozdíl. Jedná se totiž "pouze" o počet fyzických cest propojení. Z praktického hlediska je však zřejmé, že mnohem výhodnější je, je-li těchto cest co nejméně.

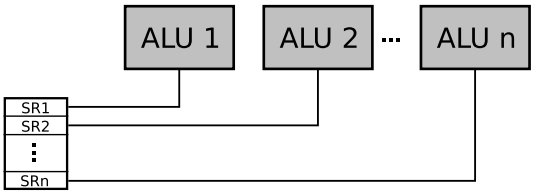
Celý systém může být propojen paralelní sběrnicí (viz obrázek 3.14).



Obrázek 3.14: Propojení paralelní sběrnicí

V každém hodinovém taktu jsou nahrána do jednoho procesoru všechna data. V dalším taktu do dalšího, atd. Tzn. do n procesorů budou data nahrána během n taktů hodin.

Paralelní sběrnici můžeme nahradit n sériovými sběrnicemi (viz obrázek 3.15), kde n udává počet mikroprocesorů.



Obrázek 3.15: Propojení sériovými sběrnicemi

Do každého procesoru současně se nahrává s každým taktem hodin bit požadovaných dat (počáteční podmínka h, integrační krok y_0) z bufferů (posuvných registrů SR1 ... SRn), kde jsou požadované hodnoty uloženy. Přenos trvá N taktů, kde N je počet bitů.

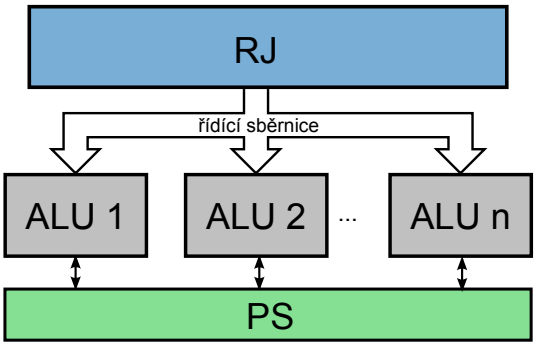
Jakmile bude počet procesorů větší než šířka potřebné paralelní sběrnice (což je náš cíl), bude sériový přenos proveden rychleji než paralelní přenos dat.

3.5 Specializovaný paralelní systém

Podoba celého vyvíjeného specializovaného paralelního systému je popsána v této kapitole. Blokové schéma, které ukazuje základní návrh tohoto paralelního systému, je na obrázku 3.16.

Celý specializovaný paralelní systém je sestaven z těchto základních bloků:

- aritmeticko-logické jednotky ALU1 ... ALUn
- propojovací systém PS
- řídící jednotky RJ



Obrázek 3.16: Blokové schéma specializovaného paralelního systému

3.5.1 Aritmeticko-logické jednotky - ALU

Aritmeticko-logické jednotky (integrátory) realizují numerickou integraci. Popis celkové koncepce je uveden výše.

Základním způsobem komunikace mezi jednotlivými elementárními mikroprocesory při výpočtu je komunikace přes sériový vstup a sériový výstup každého procesoru - preferovaný typ integrátoru je sériově-paralelní varianta.

3.5.2 Řídící jednotka - RJ

Řízení celého paralelního systému je realizováno řídící jednotkou RJ.
Základní úkoly jednotky jsou:

- řízení zápisu dat do všech výpočetních jednotek
- řízení výpočtu paralelního systému
- řízení distribuce výsledku

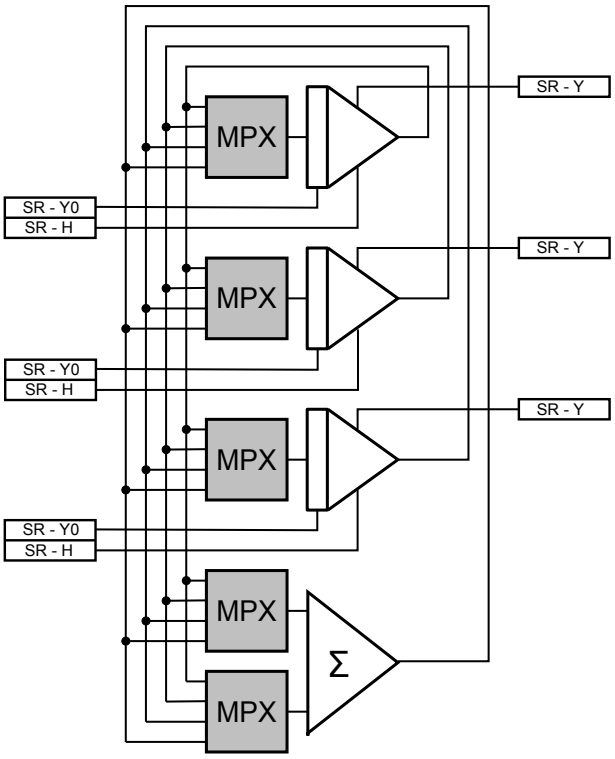
Řídící jednotka přejímá řízení, případně spolupracuje s nadřazenou jednotkou - počítačem. Nadřazená jednotka se stará o transformaci vstupních diferenciálních rovnic s využitím tvořících diferenciálních rovnic. Podle nově vzniklých diferenciálních rovnic provede propojení paralelního systému - nastavení propojovacího systému. Dále zajistí počáteční data (y_0 a h) a předá řízení řídící jednotce. Ta se stará o zápis dat, řídí výpočet a distribuci výsledku.

3.5.3 Propojovací systém - PS

Propojovací systém slouží k propojení vstupů a výstupů jednotlivých výpočetních bloků (integrátorů, sčítaček, násobiček) mezi sebou podle řešené soustavy diferenciálních rovnic. Dále musí sloužit k nahrání počátečních dat (počáteční podmínky a integrační kroky) a vypočteného výsledku. V neposlední řadě musí být každý integrátor připojen řídící sběrnicí na řídící jednotku.

Na obrázku 3.17 je detailnější blokové schéma paralelního systému zobrazující zejména propojovací systém. Tento paralelní systém obsahuje 3 integrátory a jednu sčítačku.

K propojení výpočetních jednotek je použit křížový přepínač. Křížový přepínač je realizován multiplexory. Každý multiplexor obsahuje registr adresy, který určuje vstupní kanál multiplexoru, který se bude dostávat na výstup - vstup připojené výpočetní jednotky.



Obrázek 3.17: Detailnější schéma paralelního systému

K nahrání počátečních dat slouží sériové sběrnice. Data jsou uložena v posuvných registrech SR - Y0 a SR - H a postupně se sériově nahrají do integrátorů. Tato verze obsahuje zvláštní sběrnici pro počáteční podmínku a integrační krok. Je možná úprava, která spočívá v použití jedné sdílené sběrnice. Napřed by se do integrátoru nahrála počáteční podmínka a následně integrační krok (nutný poloviční počet cest, ale pomalejší nahrávání). A nakonec systém obsahuje sériové sběrnice pro nahrání vypočtených výsledků z integrátorů do posuvných registrů SR - Y.

3.6 Shrnutí

Z rozboru provedeného v této kapitole vyplývá, že specializovaný paralelní systém bude typu SIMD. Všechny výpočetní jednotky provádí stejný výpočet nad různými daty a mohou být řízeny z jedné řídící jednotky.

Aritmeticko-logická jednotka provádí pouze dvě základní operace: násobení a sčítání. ALU představuje integrátor a tvoří základ celého paralelního systému. Výpočet těchto dvou základních operací a způsob komunikace může být prováděn sériově nebo paralelně. Z toho vyplývá rozdělění integrátorů na paralelně-paralelní, sériově-paralelní a sériově-sériové.

Dalšími neméně důležitými částmi paralelního systému jsou řídící jednotka a propojovací síť. K řízení není potřeba použít univerzální řadič, protože je pevně dán sled operací (instrukcí), a je proto použit specializovaný řadič. Je nutno navrhnout dva typy propojovacích sítí. Jedna pro propojení všech výpočetních jednotek mezi sebou a druhá pro nahrání počátečních dat a výsledků. Preferovanou variantou je sériové propojení, které zabírá podstatně menší plochu čipu a dovolí vyšší přenosovou rychlost.

Kapitola 4

Implementace a porovnání specializovaného paralelního systému

Tato kapitola obsahuje detailnější popis technické realizace celého paralelního systému, jehož návrh je představen v předchozí kapitole a je zde také provedeno srovnání implementovaných paralelních systémů.

4.1 Technická realizace

Byl navržen a realizován celý paralelní systém, společně se všemi verzemi aritmeticko-logických jednotek (integrátorů) v pevné řádové čárce. Implementace proběhla v jazyce VHDL, který slouží pro popis hardware. Detailnější informace o tomto jazyku, ze kterých jsem čerpal při své práci, jsou v [Dou03], [Hwa06]. Vytvořený systém byl odsimulován (simulátor VHDL - *ModelSim* [Gra12]) a následně byla provedena syntéza a zápis do hradlových polích FPGA.

V průběhu prací na projektu se objevil požadavek školitele na implementaci algoritmu Taylorovy řady do školního přípravku FTTkit [FIT11] a zařazení tohoto přípravku do výuky předmětu Prvky počítačů (IPR). Z těchto důvodů je výrazná část impelemtace věnována využití přípravku FITkit.

4.1.1 Porovnání jednoprocesorových systémů

Následuje srovnání implementace paralelních systémů obsahující jednotlivé varianty integrátorů. Sledovanými parametry byla obsazenost čipu a dosažitelná rychlost výpočtu v závislosti na použité datové šířce operandů.

Ve všech případech systém řešil homogenní lineární diferenciální rovnice 1. řádu s konstantními koeficienty - rovnice (2.13), blokově je tato rovnice znázorněna na obrázku 2.1.

Implementované systémy obsahují:

- integrátor
- řadič tvořený Moorovým automatem (automat řídí činnost popsanou vývojovými diagrami uvedenými u každé aritmeticko-logické jednotky), počet členů Taylorovy řady nastaven na 8

- vnitřní paměť pro 8 podílů integračního kroku
- statickou propojovací síť
- řadič komunikačního systému FITkitu - SPI

Závislosti obsazení plochy čipu a rychlosti výpočtu na šířce dat pro jednotlivé verze paralelních systémů jsou v tabulke 4.1. V tabulce je také zobrazena doba násobení násobičky u paralelně-paralelní verze a u všech verzí je uvedena doba potřebná k operaci sčítání - paralelní sčítačkou u sériově-paralelní verze a sériovou (jednobitovou) sčítačkou u sériově-sériové verze systému. Doba výpočtu ukazuje jak dlouho trvá výpočet jednoho členu Taylorovy řady DYp_i. Doba výpočtu je získána dosazením do rovnic (3.1), (3.2) a (3.3).

Tabulka 4.1: Implementace specializovaného paralelního systému systému

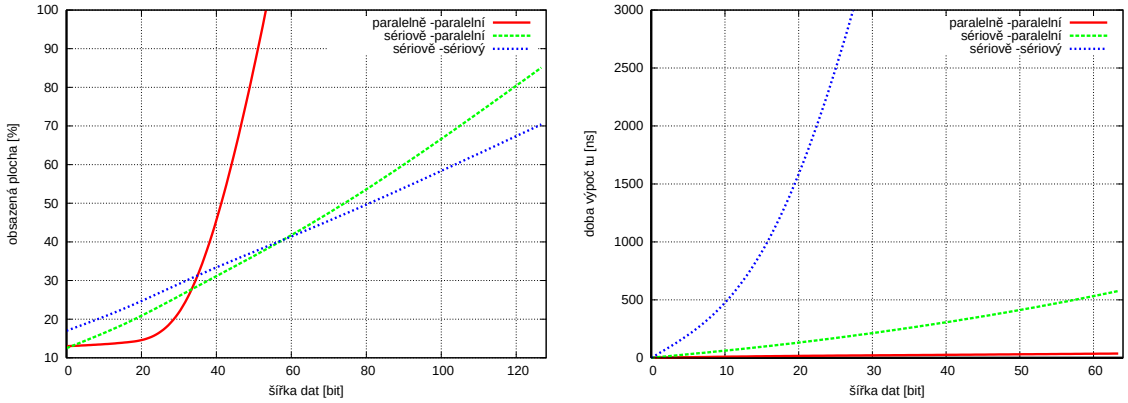
šířka dat [bit]	**16**	**32**	**64**	**16**	**32**	**64**	**128**	**16**	**32**	**64**	**128**
obsazená plocha [%]	14	25	154	19	27	44	86	23	30	43	71
doba násobení [ns]	8	15	28	96	224	576	1664	1024	4096	16384	65536
doba sčítání [ns]	6	7	9	6	7	9	13	4	4	4	4
doba výpočtu [ns]	14	22	37	102	231	585	1677	1088	4224	16640	66048
typ integrátoru	**paralelní**			**sériově-paralelní**				**sériově-sériový**			

První testovaný výpočetní systém obsahuje paralelně-paralelní verzi integrátoru. Základem je paralelní násobička, která vznikla automatickou syntézou, protože čip neobsahuje integrované násobičky. Integrátor je propojen paralelním přímým propojením a data jsou do něho nahrána pomocí paralelní sběrnice.

Druhý výpočetní systém je založen na sériově-paralelní verzi integrátoru. Integrátor je propojen sériově a data jsou do něho nahrána také přes paralelní sběrnici jako v předchozí variantě.

Jádro posledního výpočetního systému obsahuje sériově-sériovou verzi integrátoru. Integrátor je propojen sériově jako v sériově-paralelní verzi a data jsou do něho nahrána sériovými sběrnicemi.

Graf závisloti obsazení plochy čipu na šířce dat a doba provedení výpočtu jednoho členu Taylorovy řady jednotlivými systémy je na obrázku 4.1.



Obrázek 4.1: Obsazenost plochy a doba výpočtu implementovaných paralelních systémů

Jak je vidět, sériově-sériová varianta má očekávaný malý nárůst velikosti při nárůstu šířky slova, ale rychlost výpočetního systému je velmi malá. Největší výhodou této varianty je, že zapojení aritmetické jednotky nezávisí na šířce slova. Nárůst obsazení plochy je tedy způsoben pouze rozšířením vnitřních registrů.

V případě sériově-paralelní varianty je nárůst plochy oproti sériově-sériové variantě malý, protože obsahuje navíc jen paralelní sčítačku a proto poměr cena/výkon se zdá být nejlepší.

Paraleně-paralení verze najde uplatnění jen v případě nutnosti velmi rychlého výpočtu a při použití čipu, který již obsahuje integrované násobičky. Nebude tedy nutné provádět syntézu násobiček a zabírat tak podstatnou část plochy čipu, která půjde využít na zbytek systému. Velká obsazenost plochy čipu je také dána paralelním propojením mezi jednotkami, které při větším počtu jednotek bude velmi náročné.

4.1.2 Porovnání víceprocesorových systémů

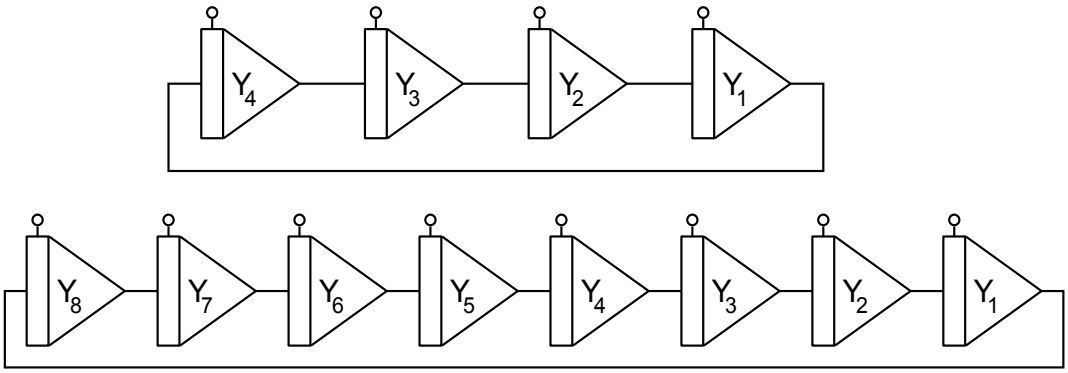
Pokud chceme jednotlivé systémy vzájemně porovnat, musíme tak učinit při řešení stejných soustav diferenciálních rovnic. Uvažujme jednoduché výpočetní schéma, které můžeme dále paralelizovat rozdělením práce mezi dalších N integrátorů, čímž by se čas potřebný pro výpočet zkrátil v ideálním případě Nkrát.

Ke srovnání jednotlivých verzí paralelních systémů mezi sebou použijeme řešení soustavy diferenciálních rovnic (4.2)–(4.5), která vznikne aplikací tvořících diferenciálních rovnic na rovnici 4.1.

$$y = \sin(t) \tag{4.1}$$

$$\begin{aligned}
y_1' &= y_2 \qquad y_1(0) = 0 && (4.2)\\
y_2' &= y_3 \qquad y_2(0) = 1 && (4.3)\\
y_3' &= y_4 \qquad y_3(0) = 0 && (4.4)\\
y_4' &= y_1 \qquad y_4(0) = -1 && (4.5)
\end{aligned}$$

Tato soustava se dá dále rozšiřovat a vznikne tak soustava 8, 16, 32,... rovnic. Odpovídající blokové schéma je na obrázku 4.2. Je zde také zobrazen princip, jak vznikne soustava osmi diferenciálních rovnic.



Obrázek 4.2: Blokové schéma řešené soustavy

Tabulka 4.2 ukazuje obsazenost čipu paralelními systémy, které obsahují 1–32 jednovstupých aritmeticko-logických jednotek různé datové šířky. Jsou zde potvrzeny výsledky z

předchozího srovnání. Navíc je vidět, že s vyším počtem integrátorů se více zvyšuje rozdíl mezi obsazeností čipu sériově-paralelní a sériově-sériovou verzí. To je způsobeno použitím paralelních sběrnic sloužícím k nahrání počátečních podmínek a integračních kroků do integrátorů.

Tabulka 4.2: Obsazenost čipu paralelním systémem s více integrátory

Počet	**šířka dat [bit]**										
ALU	**16**	**32**	**64**	**16**	**32**	**64**	**128**	**16**	**32**	**64**	**128**
1	14%	25%	154%	19%	27%	44%	86%	23%	30%	43%	71%
2	17%	99%	-	23%	37%	64%	117%	25%	33%	48%	82%
4	25%	-	-	34%	56%	101%	-	28%	38%	59%	99%
8	-	-	-	57%	95%	-	-	35%	50%	79%	-
16	-	-	-	99%	-	-	-	50%	72%	115%	-
32	-	-	-	-	-	-	-	78%	110%	-	-
	paralelně-paralelní			**sériově-paralelní**				**sériově-sériový**			

Pro srovnání výkonnosti specializovaného paralelního výpočetního systému s klasickým výpočtem na univerzálním procesoru, předpokládejme, že integrace jednoho integrátoru spotřebuje 10 taktů procesoru. V těchto zmíněných deseti taktech je obsaženo načtení vstupní hodnoty integrátoru, její zpracování, vynásobení integračním krokem, načtení předcházející hodnoty integrátoru a její sečtení s výsledkem násobení a následné uložení zpět do paměti.

Aby specializovaný systém, ve kterém se provádí numerická integrace paralelně ve všech integrátorech, byl stejně rychlý jako výpočet na univerzálním procesoru, musí být v systému tolik integrátorů, aby platil následující vztah

$$N = \frac{t_S}{10 \cdot t_P}$$

Kde t_S je perioda výpočtu specializovaného systému a t_P perioda hodin univerzálního procesoru. Např. při použití sériově-paralelního 32bitového systému a procesoru s taktovacím kmitočtem 1 GHz bude výpočet u obou stejně rychlý při použití 23 integrátorů.

4.2 Shrnutí

V této kapitole jsem se zaměřil na detailnější popis implementace paralelního systému ve variantách paralelně-paralelní, sériově-paralelní a sériově-sériové. Implementace byla proveda v jazyce VHDL a následně byla provedena syntéza do hradlových polí FPGA na přípravku FITkit.

Na základě implementace bylo provedeno srovnání paralelních systémů obahující jednotlivé variaty aritmeticko-logických jednotek. Sledovanými parametry byla obsazenost čipu a rychlost výpočtu v závislosti na datové šířce. Jako nejlepší se jeví použít sériově-paralelní variantu.

Kapitola 5

Závěr

Předložená disertační práce se zabývá návrhem specializovaného paralelního systému, který provádí numerický výpočet diferenciálních rovnic pomocí metody Taylorovy řady. Celá práce bude nyní shrnuta v několika krocích. Nejprve bude představen použitý přístup k řešení, dále dosažené výsledky a konečně naznačení možností dalšího výzkumu.

5.1 Zvolený přístup k řešení

Při simulaci spojitých systémů se využívá popis pomocí diferenciálních rovnic. Vzniklé diferenciální rovnice mohou být převedeny na blokový diagram, který je chápán jako pralelně pracující systém. Jednotlivé prvky blokového diagramu (sčítačky, násobičky, integrátory) jsou implementovány digitálními diskrétními ekvivalenty. Díky flexibilitě a možnosti rekonfigurace jsou pro implementaci použity čipy FPGA. Základní stavební prvek celého systému je integrátor, který provádí numerickou integrací pomocí metody Taylorovy řady. Důležitou náplní této disertační práce je proto i numerická integrace. Bylo dokázáno, že metoda se Taylorovy řady vyznačuje nejek vysokou přesností výpočtu, ale především vysokým stupněm paralelismu.

Díky metodice tvořících diferenciálních rovnic, která je popsána v této práci, lze metodou Taylorovy řady řešit soustavy homogenních diferenciálních rovnic, nehomogenních diferenciálních rovnic, diferenciálních rovnic s proměnnými koeficienty a soustavy nelineárních diferenciálních rovnic.

Bylo provedeno srovnání efektovnosti metody Taylorovy řady a obecně velmi používanými metodami Runge-Kutta druhého a čtvrtého řádu. Ze srovnání vyplývá, že při výpočtu pomocí Taylorovy řady je počet výpočetních operací menší než metodou Runge-Kutta odpovídajícího řádu. S použitím vyššího řádu je tento rozdíl ještě výraznější.

Všechny výpočetní jednotky provádí stejný výpočet a jsou řízeny z jedné řídící jednotky. Proto byla zvolena paralelní architektura typu SIMD. Podle způsobu provádění výpočetních operací v aritmeticko-logických jednotkách a způsobu komunikace mezi nimi, se rozdělují na paralelně-paralelní, sériově-paralelní a sériově-sériové. Je použito uložení čísel v pevné řádové čárce, protože výpočty v této aritmetice jsou rychlejší a mají menší nároky na hardwarové zdroje.

5.2 Dosažené výsledky

Specializovaný paralelní systém byl implementován na přípravku FITkit, který obsahuje FPGA čip typu Spartan 3 od společnosti Xilinx. Bylo provedeno srovnání implementovaných paralelních systémů využívající jednotlivé varianty integrátorů. Sledovanými parametry byla obsazenost čipu a rychlost výpočtu v závislosti na datové šířce. Jako nejlepší se jeví použít sériově-paralelní variantu.

Největší výhoda této varianty je sériové propojení. Tento typ propojení umožňuje rychlejší přenos dat a několikanásobně šetří použité prvky na FPGA čipu oproti paralelní variantě. Propojovací síť je příliš náročná. Nevýhoda paralelní varianty je i nutnost realizovat násobičku, která je nejsložitější prvek celé výpočetní jednotky. Tento nedostatek ale odpadá při použití výkonnějšího FPGA čipu, který již obsahuje integrované násobičky.

Využití paralelního systému je ukázáno na skutečné výukové aplikaci, která vznikla jako podpora laboratorní výuky předmětu prvky počítačů. Další možnou aplikací je teorie řízení. Reálný systém je řízen modelem, jehož chování by mělo být shodné s modelovaným reálným systémem. Model je založen na popisu pomocí Taylorovy řady.

5.3 Možnosti dalšího výzkumu

Tato disertační práce si kladla za cíl podrobně popsat možnosti, jak urychlit úspěšnou metody Taylorovy řady pomocí paralelního hardwarového zpracování. Žádná podobná práce tohoto druhu zatím nevznikla a proto by měla sloužit jako podklad pro další výzkum a vylepšení v této oblasti. V této chvíli se nabízí několik možností, na co se dále zaměřit.

Implemntovat rozsáhlejší paralelní systém na výkonnější architektuře než FITkit - např FPGA čip typu Virtex a provést detailní srovnání se systémy, které řeší podobné problémy. Tímto se mohou potvrdit dosavadní příznivé výsledky a může následně pokračovat další výzkum.

Jak bylo v této práci několikrát zmíněno, velmi důležitou částí každého paralelního systému je propojovací síť. Další výzkum může být tedy proveden v oblasti propojovacích sítí - navrhnout několik variant, důkladně analyzovat a provést podrobné srovnání.

Při řešení tuhých systémů dochází k velkému rozptylu vstupních i počítaných hodnot. Zde představená koncepce v pevné řádové čárce by proto nebyla dostačující pro řešení takovýchto problémů. Nabízí se proto dále rozšířit koncepce integrátoru v provedení pohyblivé řádové čárky. Další možností je navrhnout a implementovat víceslovní aritmetiku, kde by se pro uložení čísel využívaly stovky bitů.

V neposlední řadě je možné navrhnout programové prostředí, které bude schopno zadaný problém transformovat pro řešení v našem parallení systému. Tzn. aplikuje tvořící diferenciální rovnice, podle toho provede celkové propojení systému a nastartuje výpočet. Následně zpracuje výsledná data a uloží je nebo zobrazí v nějaké uživatelsky přívětivé formě (např. do grafu).

Literatura

[BHSL82] M. Bobek, J. Haška, I. Serba, and M Lukeš. *Analogové počítače*. SNTL Praha, 1982. ISBN 0451082.

[But00] J. C. Butcher. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*, 125(1-2):1–29, 2000. ISSN 0377-0427.

[But08] J. C. Butcher. *Numerical methods for ordinary differential equations*. Wiley, 2 edition, 2008. ISBN 978-0-470-72335-7.

[BWZ72] D. Barton, I. M. Willers, and R. V. M. Zahar. Taylor series methods for ordinary differential equations - an evaluation. In *Mathematical Software Symposium*, pages 369–390. Ed. Academic Press, New York, 1972.

[DB91] J. Diblík and J. Baštinec. *Matematika III*. VUT v Brně, 1 edition, 1991. ISBN 80-214-0315-2.

[Dou03] J. Douša. *Jazyk VHDL*. České vysoké učení technické v Praze, vydavatelství ČVUT, 2003. ISBN 80-01-02670-1.

[Drá95] V. Drábek. *Výstavba počítačů*. Vysoké učení technické v Brně, nakladatelství PC-DIR, 1995. ISBN 80-214-0691-7.

[DT04] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 0-12-200751-4.

[Dvo04] V. Dvořák. *Architektura a programování parelelních systémů*. Vysoké učení technické v Brně, nakladatelství VUTIUM, 2004. ISBN 80-214-2608-X.

[FIT11] FIT. Domovská web stránka fit-kitu [online], [cit. 2-5-2011]. `<http://merlin.fit.vutbr.cz/FITkit/>`.

[Gib60] A. Gibbons. A program for the automatic integration of differential equations using the method of Taylor series. *Computer Journal*, 3(5):108–111, 1960.

[Gra12] Mentor Graphics. High performance and capacity mixed HDL simulation - ModelSim [online], [cit. 2-3-2012]. `<http://www.mentor.com/products/fv/modelsim/>`.

[Hol99] O. Holub. Numerické řešení rozsáhlých soustav diferenciálních a algebraických rovnic. Master's thesis, FEI, VUT, 1999.

[HW96] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II.* Springer series in computational mathematics. Springer-Verlag, 1996. ISBN 3-540-60452-9.

[Hwa06] Enoch O. Hwang. *Digital Logic and Microprocessor Design with VHDL.* Thomson, 1 edition, 2006. ISBN 0-534-46593-5.

[KDJ05] M. Kubíček, M. Dubcová, and D. Janovská. *Numerické metody a algoritmy.* Vydavatelství VŠCHT Praha, 2 edition, 2005. ISBN 80-7080-558-7.

[Kun94] J. Kunovský. *Modern Taylor Series Method.* FEI-VUT Brno, 1994. Habilitation work.

[Mac07] B. J. MacLennan. A review of analog computing. Technical report, Department of Electrical Engineering & Computer Science University of Tennessee, Knoxville, 2007.

[Map12] MapleSoft. Maplesoft - technical computing software [online], [cit. 10-4-2012]. `<www.maplesoft.com/products/maple/>`.

[Mat12] MathWorks. Matlab - the language of technical computing [online], [cit. 10-4-2012]. `<www.mathworks.com/products/matlab/>`.

[Mik00] K. Mikulášek. *Polynomial Transformations of Systems of Differential Equations and Their Applications.* PhD thesis, FEI-VUT Brno, 2000.

[PP06] J. Pinker and M. Poupa. *Číslicové systémy a jazyk* VHDL. Ben - technická literatura, 1 edition, 2006. ISBN 80-7300-198-5.

[Šát12] V. Šátek. *Analýza stiff soustav diferenciálních rovnic.* PhD thesis, Department of Intelligent Systems FIT BUT, 2012.

[Šva99] S. Švarc. *Matematická analýza I.* VUT v Brně, 4 edition, 1999. ISBN 80-214-1411-1.

Přehled publikací autora

[1] G. Fuchs, J. Kopřiva, M. Kraus, and M. Kozek. Application of the modern taylor series method to a multi-torsion chain. In *Proceedings of the 7th EUROSIM Congress on Modelling and Simulation*, page 7. Czech Technical University Publishing House, 2010. ISBN 978-80-01-04589-3.

[2] V. Kaluža, M. Kraus, J. Kunovský, and V. Šátek. Initial problems with polynomials on right-hand sides. In *Proceedings of Third Asia International Conference on Modelling and Simulation*, pages 182–187. IEEE Computer Society, 2009. ISBN 978-1-4244-4154-9.

[3] M. Kraus and J. Kunovský. Adapting power-series integration to real-time simulation. In *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation*, page 6, 2007. ISBN 978-3-901608-32-2.

[4] M. Kraus, J. Kunovský, M. Pindryč, and V. Šátek. Taylor series in control theory. In *Proceedings UKSim 10th International Conference on Computer Modelling and Simulation (EUROSIM/UKSim)*, pages 378–379. IEEE Computer Society, 2008. ISBN 978-0-7695-3114-4.

[5] M. Kraus, J. Kunovský, and V. Šátek. Fourier analysis and modern taylor series method. In *Proceedings of the 6th International Conference on APLIMAT*, pages 203–208, 2007. ISBN 978-80-969562-5-8.

[6] M. Kraus, J. Kunovský, and V. Šátek. Taylor series numerical integrator. In *Second UKSIM European Symposium on Computer Modeling and Simulation (EMS 2008)*, pages 177–180. IEEE Computer Society, 2008. ISBN 978-0-7695-3325-4.

[7] M. Kraus, J. Kunovský, and V. Šátek. Taylorian initial problem. In *Proceedings MATHMOD 09 Vienna - Full Papers CD Volume*, pages 1181–1186. ARGE Simulation News, 2009. ISBN 978-3-901608-35-3.

[8] J. Kunovský, J. Konvalina, and M. Kraus. Implementace TKSL v pevné a pohyblivé řádové čárce. In *Proceedings of 40th Spring International Conference Mosis'06, Modelling and Simulation of Systems*, pages 259–264, 2006. ISBN 80-86840-21-2.

[9] J. Kunovský, M. Kraus, V. Šimek, and J. Petřek. GPU based accelleration of telegraph equation. In *Proceedings UKSim 10th International Conference on Computer Modelling and Simulation (EUROSIM/UKSim)*, pages 629–630. IEEE Computer Society, 2008. ISBN 978-0-7695-3114-4.

[10] J. Kunovský, M. Kraus, V Šátek, and V. Kaluža. Accuracy and word width in TKSL. In *Second UKSIM European Symposium on Computer Modeling and Simulation (EMS 2008)*, pages 153–158. IEEE Computer Society, 2008. ISBN 978-0-7695-3325-4.

[11] J. Kunovský, M. Kraus, V. Šátek, and A. Szöllös. Parallel computations based on modified numerical integration methods. In *Proceedings of the 10th International Conference of Numerical Analysis and Applied Mathematics*, volume 1479, pages 2217–2220. American Institute of Physics, 2012. ISBN 978-0-7354-1091-6.

[12] J. Kunovský, M. Kraus, and V. Vopěnka. Runge-kutta based parallel computations. In *Proceedings of the MATHMOD VIENNA 2012 - 7th Vienna Conference on Mathematical Modelling*, page 6. ARGE Simulation News, 2012.

[13] J. Kunovský, V. Šátek, and M. Kraus. 25th anniversary of TKSL. In *Numerical Analysis and Applied Mathematics*, volume 1048, pages 343–346. American Institute of Physics, 2008. ISBN 978-0-7354-0576-9.

[14] J. Kunovský, V. Šátek, and M. Kraus. New trends in taylor series based computations. In *Numerical Analysis and Applied Mathematics*, volume 1168, pages 282–285. American Institute of Physics, 2009. ISBN 978-0-7354-0709-1.

[15] J. Kunovský, V Šátek, M. Kraus, and J. Kopřiva. Semi-analytical computations based on TKSL. In *Second UKSIM European Symposium on Computer Modeling and Simulation (EMS 2008)*, pages 159–164. IEEE Computer Society, 2008. ISBN 978-0-7695-3325-4.

[16] J. Kunovský, V. Šátek, M. Kraus, and J. Kopřiva. Automatic method order settings. In *Proceedings of Eleventh International Conference on Computer Modelling and Simulation (UKSim 2009)*, pages 117–122. IEEE Computer Society, 2009. ISBN 978-1-4244-3771-9.

[17] J. Kunovský, V. Šátek, M. Kraus, and M. Pindryč. Comparison of TKSL to world standards. *International Journal of Autonomic Computing*, 1(2):182–191, 2009. ISSN 1741-8569.

[18] V. Kunovský, M. Kraus, V. Šátek, and V. Kaluža. Parallel computations based on analogue principles. In *Proceedings of Eleventh International Conference on Computer Modelling and Simulation (UKSim 2009)*, pages 111–116. IEEE Computer Society, 2009. ISBN 978-1-4244-3771-9.

[19] V. Šátek, J. Kunovský, M. Kraus, and M. Pindryč. TKSL to world standards comparison. In *MOSIS '08*, pages 21–27, 2008. ISBN 978-80-86840-40-6.

Životopis

Osobní informace

Jméno	Michal Kraus
Datum narození	7. 10. 1981
Adresa	Na Zahrádkách 921, 664 53 Újezd u Brna, Česká republika
Email	m.kraus@seznam.cz

Vzdělání

2006 – dodnes	Vysoké učení technické v Brně, Fakulta informačních technologií **Doktorský studijní program**, Výpočetní technika a informatika Téma disertační práce: *Paralelní výpočetní architektury založené na numerické integraci*
2001 – 2006	Vysoké učení technické v Brně, Fakulta informačních technologií **Magisterský studijní obor**, Informatika a výpočetní technika Téma diplomové práce: *Elementární procesor specializovaného paralelního systému*

Zaměstnání

2011 – dodnes	**Software design engineer**, Honeywell Komunikační jednotka letadel - vývoj, testování, dokumentace a certifikační aktivity splňující normu DO-178B a FAA Order 8110.49 Software Approval Guidelines
2010 – 2011	**Programátor**, Seznam.cz a.s. Vývoj serverových aplikací v jazyce C/C++
2007 – 2010	**Asistent na Ústavu inteligentních systémů**, Vysoké učení technické v Brně, Fakulta informačních technologií Aktivity spojené s výukou, vědecká a výzkumná činnost v oblasti disertační práce