

Multimediální podpora výuky spojitého řízení

Martin Boráň

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Martin Boráň
Osobní číslo: A22158
Studijní program: B0714A150006 Aplikovaná informatika v průmyslové automatizaci
Specializace: Inteligentní systémy s roboty
Forma studia: Kombinovaná
Téma práce: Multimediální podpora výuky spojitého řízení
Téma práce anglicky: Multimedia support for teaching continuous control

Zásady pro vypracování

- Vypracujte literární rešerši z oblasti základů spojitého řízení a MATLAB.
- Analyzujte existující podpory výuky z teorie řízení.
- Vytvořte vlastní návrh aplikace v programu MATLAB s využitím App Designer pro podporu výuky spojitého řízení.
- Popište vytvořené funkce a soubory, vytvořte vývojový diagram aplikace.
- Otestujte funkčnost aplikace a popište výsledky ověření.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PERŮTKA, Karel. MATLAB – základy pro studenty automatizace a informačních technologií. Učební texty vysokých škol. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. ISBN 80-7318-355-2.
2. DUŠEK, František. MATLAB a SIMULINK: úvod do používání. Pardubice: Univerzita Pardubice, 2000. ISBN 80-7194-273-1.
3. ZAPLATÍLEK, Karel a DOŇAR, Bohuslav. MATLAB pro začátečníky. 2. vyd. Praha: Ben – technická literatura, 2005. ISBN 80-7300-175-6.
4. BALÁTĚ, Jaroslav. Automatické řízení. 2., přeprac. vyd. Praha: BEN, 2004. ISBN 80-7300-148-9.
5. VAŠEK, V. Teorie automatického řízení II. 1., Brno: Rektorát Vysokého učení technického v Brně, 1990. ISBN 80-214-0115-X.
6. PROKOP, Roman; MATUŠŮ, Radek a PROKOPOVÁ, Zdenka. Teorie automatického řízení – lineární spojité dynamické systémy. Učební texty vysokých škol. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006. ISBN 80-7318-369-2.

Vedoucí bakalářské práce: **Ing. Karel Perůtka, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **6. prosince 2024**
Termín odevzdání bakalářské práce: **27. května 2025**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan



prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 6. prosince 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Bakalářská práce se zabývá vývojem výukové aplikace v prostředí MATLAB pomocí nástroje App Designer. Výsledná aplikace slouží k procvičování znalostí spojitého řízení. Teoretická část se věnuje základům MATLABu, spojitého řízení a přehledu podobných výukových aplikací. Praktická část popisuje návrh a tvorbu aplikace z pohledu programátora i uživatele.

Klíčová slova: Spojité řízení, MATLAB, App Designer, Automatizace, Programování, Hra, Výuková aplikace

ABSTRACT

The bachelor thesis focuses on the development of an educational application in the MATLAB environment using the App Designer tool. The resulting application is intended for practicing knowledge of continuous control. The theoretical part covers the basics of MATLAB, continuous control, and provides an overview of similar educational applications. The practical part describes the design and development of the application from both a programmer's and a user's perspective.

Keywords: Continuous control, MATLAB, App Designer, Automation, Programming, Game, Educational application

Rád bych tímto poděkoval vedoucímu bakalářské práce, panu Ing. Karlu Perůtkovi, Ph.D., za jeho vstřícný přístup, cenné rady, trpělivost a odborné vedení, které mi velmi pomohly při tvorbě této práce.

Velké díky patří také mé rodině a blízkým přátelům za jejich neustálou podporu, trpělivost a motivaci v průběhu celého studia. Bez jejich zázemí a povzbuzení by cesta k dokončení této práce byla výrazně složitější.

OBSAH

ÚVOD	13
I TEORETICKÁ ČÁST	14
1 MATLAB.....	15
1.1 O PROSTŘEDÍ MATLAB.....	15
1.1.1 Vlastnosti	15
1.2 ROZHRANÍ APLIKACE.....	16
1.2.1 Toolstrip (Nástrojová lišta)	16
1.2.2 Current Folder (Aktuální složka)	17
1.2.3 Details (Detaily)	17
1.2.4 Command Window (Příkazové okno)	17
1.2.5 Workspace (Pracovní prostor).....	17
1.3 PROMĚNNÉ.....	17
1.4 OPERÁTORY	18
1.4.1 Relační operátory.....	18
1.4.2 Logické operátory.....	19
1.5 TVORBA SCRIPTŮ	19
1.6 FUNKCE	19
1.7 TVORBA LIVE SCRIPTŮ.....	19
1.8 ŘÍDICÍ STRUKTURY	20
1.8.1 Podmínka if/else	20
1.8.2 Switch	20
1.8.3 Cyklus for.....	21
1.8.4 Cyklus while.....	21
1.9 ZÁKLADNÍ PŘÍKAZY.....	22
1.9.1 Správa prostředí a proměnných.....	22
1.9.2 Práce s maticemi a vektory	22
1.9.3 Práce s čísly.....	23
1.9.4 Práce s řetězci.....	23
1.9.5 Práce s grafy.....	24
1.10 INTERAKTIVNÍ EDITACE GRAFŮ	24
1.11 APP DESIGNER	25
1.11.1 Prostředí App Designeru.....	25
1.11.2 Programování	27
2 ZÁKLADY SPOJITÉHO ŘÍZENÍ.....	28
2.1 OVLÁDÁNÍ A REGULACE	28
2.2 REGULAČNÍ OBVOD	29
2.3 SPOJITÉ A NESPOJITÉ ŘÍZENÍ.....	30

2.4	LAPLACEOVA TRANSFORMACE	30
2.4.1	Přímá Laplaceova transformace	31
2.4.2	Zpětná Laplaceova transformace	32
2.4.3	Podmínky použití Laplaceovy transformace	32
2.4.4	Heavisideův rozvoj	32
2.5	POPIS SYSTÉMU LINEÁRNÍ DIFERENCIÁLNÍ ROVNICÍ	33
2.6	PŘENOS SYSTÉMU	33
2.7	NULY A PÓLY SYSTÉMU	34
2.8	PŘECHODOVÁ FUNKCE	34
2.9	IMPULZNÍ FUNKCE	34
2.10	FREKVENČNÍ PŘENOS	35
2.11	AMPLITUDO-FÁZOVÁ FREKVENČNÍ CHARAKTERISTIKA V KOMPLEXNÍ ROVINĚ	35
2.12	FREKVENČNÍ CHARAKTERISTIKA V LOGARITMICKÝCH SOUŘADNICÍCH	36
2.13	REGULÁTOR	36
2.13.1	Členění regulátorů	36
2.13.2	Spojité regulátory	37
2.13.3	Realizace spojitých regulátorů	37
2.14	REGULOVANÁ SOUSTAVA	40
2.15	TYPY REGULOVANÝCH SOUSTAV	40
2.15.1	Proporcionální (statické) regulované soustavy	40
2.15.2	Integrační (astatické) regulované soustavy	43
2.15.3	Regulované soustavy s dopravním zpožděním	45
2.16	STABILITA SYSTÉMU	46
2.17	KRITÉRIA STABILITY	47
2.17.1	Algebraická kritéria stability	47
2.17.2	Routh-Schurovo kritérium stability	48
2.17.3	Hurwitzovo kritérium stability	49
2.17.4	Geometrická kritéria stability	49
2.17.5	Michajlovo-Leonhardovo kritérium stability	50
2.17.6	Nyquistovo kritérium stability	51
3	ANALÝZA EXISTUJÍCÍCH PODPOR VÝUKY TEORIE ŘÍZENÍ	53
3.1	NÁVRH APLIKACE A SADY PŘÍKLADŮ PRO PODPORU VÝUKY AUTOMATIZACE PRO PREZENČNÍ A KOMBINOVANÉ STUDIUM	53
3.2	PODPORA CVIČENÍ Z PŘEDMĚTU: TEORIE AUTOMATICKÉHO ŘÍZENÍ I	54
3.3	GRAFICKÝ NÁSTROJ PRO PODPORU VÝUKY TEORIE SYSTÉMŮ A ŘÍZENÍ	55
3.4	VÝUKOVÉ ANIMACE PRO PŘEDMĚT SPOJITÉ ŘÍZENÍ	56
3.5	MULTIMEDIÁLNÍ PODPORA VÝUKY PŘEDMĚTU TAŘ II – SYNTÉZA	57
II	PRAKTICKÁ ČÁST	59
4	SPECIFIKACE APLIKACE	60

4.1	POPIS APLIKACE	60
4.2	PRAVIDLA HRY	60
5	VÝVOJ APLIKACE	61
5.1	PROCES TVORBY APLIKACE.....	61
5.2	STRUKTURA APLIKACE	61
5.2.1	Menu.mlapp	62
5.2.2	Pravidla.mlapp.....	65
5.2.3	Hra.mlapp.....	66
5.3	KLÍČOVÉ FUNKCE A HERNÍ LOGIKA	68
5.3.1	startupFcn.....	68
5.3.2	centerFigure.m.....	69
5.3.3	SetupImage.....	70
5.3.4	LoadQuestions.....	71
5.3.5	SetupButtons	72
5.3.6	UpdateMarioPosition	73
5.3.7	MarioClicked.....	73
5.3.8	FieldClicked	74
5.3.9	OdpovdtButtonPushed	76
5.3.10	CorrectAnswer.....	78
5.3.11	WrongAnswer	81
5.3.12	RESETButtonPushed.....	82
5.3.13	ZPTButtonPushed a EXITButtonPushed.....	83
5.4	DATOVÉ SOUBORY	83
5.4.1	questions.xlsx	83
5.4.2	leaderboard.mat	85
5.5	POHLED ZE STRANY UŽIVATELE	86
5.5.1	Spuštění aplikace	86
5.5.2	Zobrazení pravidel.....	87
5.5.3	Tabulka výsledků.....	88
5.5.4	Zahájení hry	89
5.5.5	Odpovídání na otázky	90
5.5.6	Vyhodnocení odpovědi	91
5.5.7	Postup hrou.....	92
5.5.8	Dokončení hry	93
5.5.9	Tabulka výsledků.....	94
5.6	TESTOVÁNÍ APLIKACE	94
5.7	ZHODNOCENÍ APLIKACE.....	96
ZÁVĚR		97
SEZNAM POUŽITÉ LITERATURY		99
SEZNAM OBRÁZKŮ		102
SEZNAM ZDROJOVÝCH KÓDŮ		104
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....		105

SEZNAM PŘÍLOH.....107

ÚVOD

Tato bakalářská práce se zabývá návrhem a tvorbou výukové aplikace v prostředí MATLAB, konkrétně pomocí nástroje App Designer. Výsledkem je vzdělávací aplikace s názvem Save the Princess, která propojuje prvky kvízu a deskové hry.

Cílem aplikace je umožnit studentům zábavnou formou procvičovat své znalosti z oblasti spojitého řízení a práce v MATLABu, a zároveň poskytnout učitelům nástroj pro procvičení dané tematiky

Teoretická část práce je zaměřena na popis prostředí MATLAB, jeho programovacích možností a nástroje App Designer, který umožňuje tvorbu aplikací s grafickým uživatelským rozhraním. Součástí rešerše jsou základy z oblasti spojitého řízení a analýza již existujících výukových aplikací, které se zabývají podobnými tématy.

Praktická část se věnuje návrhu, tvorbě a testování samotné aplikace. Je popsána jak z pohledu programátora, včetně struktury kódu a použitých funkcí, tak i z pohledu uživatele. Aplikace byla vytvořena s důrazem na jednoduché ovládání, jednoduchý design a možnost snadné úpravy otázek dle potřeb vyučujícího. Výsledkem je aplikace, která může pomoci studentům lépe pochopit náročnější témata zábavnou formou.

I TEORETICKÁ ČÁST

1 MATLAB

MATLAB (zkratka pro **Matrix Laboratory**) je rozsáhlý software, který obsahuje vlastní skriptovací programovací jazyk. Program byl vyvinut společností MathWorks, která jej uvedla na trh v roce 1984. Původně byl MATLAB určen pouze pro operační systém UNIX, avšak s rostoucí popularitou a vývojem technologií byl postupně rozšířen i na další operační systémy jako jsou macOS a Windows.

1.1 O prostředí MATLAB

Původně byl MATLAB určen pro matematické účely, ale časem byl rozšířen do rozsáhlého prostředí, které umožňuje složité matematické výpočty, vykreslování 2D i 3D grafů funkcí, analýzu a prezentaci dat, programování, provádění počítačových simulací či vývoj aplikací, včetně grafického uživatelského rozhraní.

1.1.1 Vlastnosti

Jeho základní datovou strukturou je matice (dvojměrné pole), což znamená, že všechny objekty v jádru MATLABu jsou vnímány jako prvky matice. Výpočetní jádro, které je základem celého prostředí, je optimalizováno pro efektivní provádění numerických operací a nabízí více než 500 vestavěných funkcí. Tyto funkce lze skládat a vytvářet tak složitější algoritmy, přizpůsobené konkrétním úkolům [1].

MATLAB nabízí také širokou škálu nástrojů, které rozšiřují jeho základní funkce. Tyto nástroje tzv. toolboxy jsou knihovny funkcí zaměřené na konkrétní oblasti.

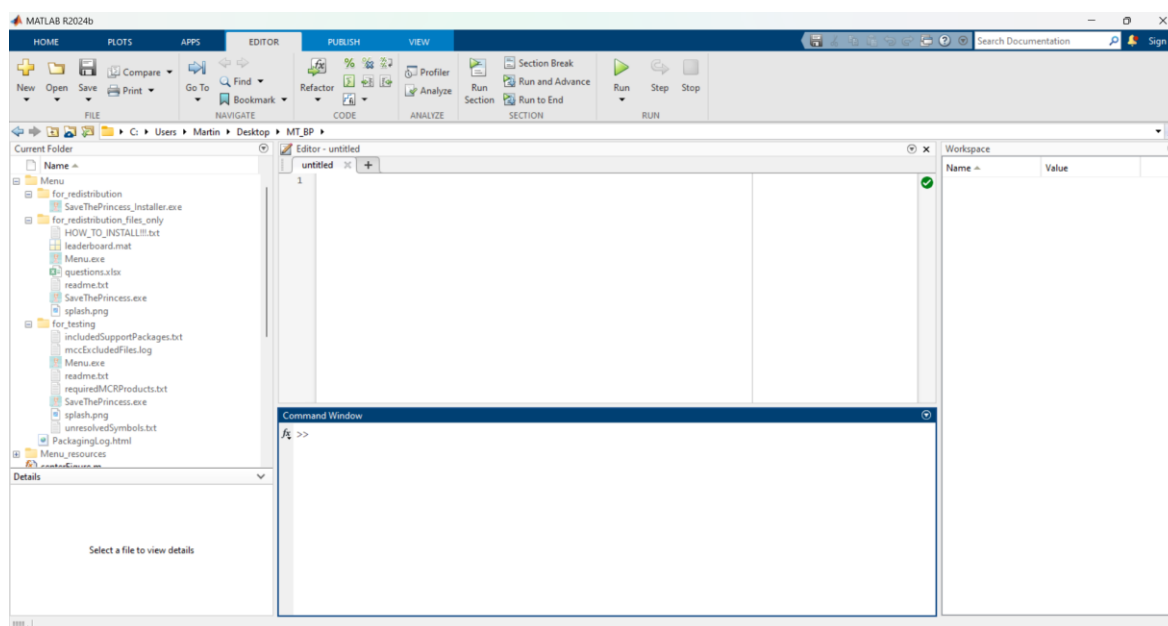
Přehled hlavních toolboxů v MATLABu [2]:

- **Symbolic Math Toolbox** – umožňuje práci se symbolickou matematikou, derivacemi, integrály, řešení rovnic apod.
- **Signal Processing Toolbox** – obsahuje funkce pro analýzu, filtrování a zpracování signálů
- **Simulink** – modelování a simulace dynamických systémů
- **Control System Toolbox** – algoritmy a nástroje pro analýzu, návrh a ladění lineárních řídicích systémů, vizualizovat jejich chování v časové a frekvenční oblasti.

- **Database Toolbox** – umožňuje snadné připojení k databázím, práci s uloženými daty a využití běžných SQL dotazů. Podporuje oblíbené databázové systémy jako MySQL, PostgreSQL, Microsoft SQL Server nebo Oracle.

1.2 Rozhraní aplikace

Po spuštění programu se zobrazí hlavní okno s uživatelským rozhraním, které je, jak lze vidět na obrázku 1, rozděleno do několika klíčových oken a panelů.



Obrázek 1 Rozhraní aplikace MATLAB po spuštění

1.2.1 Toolstrip (Nástrojová lišta)

Nástrojová lišta je umístěná v horní části okna aplikace. Po spuštění obsahuje tři hlavní záložky (HOME, PLOTS, APPS), které poskytují rychlý přístup k nejdůležitějším funkcím a nástrojům MATLABu. V případě otevření skriptu se zobrazí další tři záložky sloužící k úpravě a ladění kódu (EDITOR, PUBLISH, VIEW). MATLAB umožňuje vytvořit si také vlastní záložky, do kterých si uživatel může vložit funkce a nástroje dle svých preferencí.

HOME

V záložce HOME nalezneme sadu nástrojů sloužících k základní správě a ovládání MATLABu. Najdeme zde další podkategorie jako FILE (správa souborů), VARIABLE (správa proměnných a dat), CODE (spuštění a kontrola kódu). Záložka HOME také poskytuje přístup k Simulinku, což je nástroj sloužící k modelování a simulaci dynamických systémů.

PLOTS

Pomocí záložky PLOTS lze rychle tvořit a upravovat grafy. Nachází se zde široký výběr předdefinovaných grafů a nástrojů k jejich úpravě.

APPS

Tato záložka slouží pro přístup k aplikacím, které rozšiřují základní funkce MATLABu. Uživatelé mohou spustit předdefinované aplikace, stahovat nové nebo vytvářet vlastní pomocí App Designeru.

1.2.2 Current Folder (Aktuální složka)

Tento panel zobrazuje aktuální adresář, umožňuje rychlý přístup k jeho složkám a souborům.

1.2.3 Details (Detaily)

Slouží k zobrazení podrobných informací o aktuálně vybraném souboru.

1.2.4 Command Window (Příkazové okno)

Hlavní konzole sloužící pro zadávání příkazů a zobrazování výsledků.

1.2.5 Workspace (Pracovní prostor)

Panel zobrazuje všechny proměnné aktuálně načtené v paměti. Vidíme zde její jméno a hodnotu.

1.3 Proměnné

V MATLABu není nutné proměnné předem deklarovat, automaticky vznikají při prvním přiřazení hodnoty. Název proměnné musí začínat písmenem a může obsahovat až 31 znaků. Je důležité říct, že MATLAB je case-sensitive, tzn. rozlišuje malá a velká písmena. MATLAB primárně zpracovává dva hlavní typy proměnných: numerické a textové řetězce. Při prvním přiřazení se proměnná automaticky přizpůsobí velikosti a typu hodnoty.

Proměnné zůstávají aktivní až do ukončení MATLABu nebo jejich odstranění příkazem. Pro proměnné můžeme použít libovolný název, avšak některá jména mají již předem definovanou hodnotu. Některé z těchto proměnných jsou uvedeny níže.

- `ans` – ukládá poslední výsledek, pokud není přiřazen jiné proměnné
- `pi` – Ludolfovo číslo (zaokrouhleně 3.14)
- `i,j` – imaginární jednotky
- `Inf` – nekonečno
- `NaN` – neplatná numerická hodnota

MATLAB pracuje s lokálními a globálními proměnnými, které se liší tím, ve které části programu je lze použít. Lokální proměnná je definovaná pouze uvnitř funkce a je taky dostupná pouze uvnitř této funkce. Naopak globální proměnná je dostupná napříč všemi funkcemi a workspace. Aby mohla být proměnná sdílena mezi různými funkcemi, musí být označena jako *global* v každé funkci i v hlavním skriptu, kde se používá [1].

1.4 Operátory

MATLAB podporuje různé typy operátorů, které umožňují provádět matematické a logické operace s čísly, maticemi i logickými hodnotami [3].

1.4.1 Relační operátory

Používají se k porovnávání hodnot.

- Rovnost (`==`)
- Nerovnost (`~=`)
- Větší než (`>`)
- Menší než (`<`)
- Větší nebo rovno (`>=`)
- Menší nebo rovno (`<=`)

1.4.2 Logické operátory

Používají se k logickým podmínkám.

- Logický AND (&)
- Logický OR (|)
- Logická negace (~)

1.5 Tvorba skriptů

Skripty jsou textové soubory s příponou **.m**, které umožňují provádět sekvenční příkazy v požadovaném pořadí. MATLAB vykonává příkazy postupně od začátku do konce. Skripty jsou ideální pro automatizaci úloh, které se často opakují. Nemusíme tedy manuálně zadávat jednotlivé příkazy do příkazového řádku, což šetří čas a minimalizuje chyby. Všechny proměnné vytvořené ve skriptu jsou uloženy ve Workspace, takže zůstávají přístupné i po skončení skriptu [1].

1.6 Funkce

Funkce jsou části kódu, které slouží k provedení určité operace. Mohou přijímat vstupy a vracet výsledky. Jsou definovány pomocí klíčového slova *function* a uloženy v **.m** souborech. Funkce se v MATLABu používají k tomu, aby se stejný kód nemusel psát opakovaně. Umožňují rozdělit program na menší části, které se dají přehledně uspořádat a znovu použít. Díky tomu je celý skript lépe organizovaný a jednodušší na údržbu. Narozdíl od skriptů, které provádějí příkazy sekvenčně, se funkce spouští jako celek a pracují s lokálními proměnnými, které nejsou přístupné z vnějšku [1].

1.7 Tvorba live skriptů

Live skripty jsou interaktivní dokumenty, které umožňují zkombinovat kód, výstupy, formátovaný text, rovnice a grafiku do jednoho souboru s příponou **mlx**. Výsledky se zobrazují přímo vedle kódu, takže je snazší sledovat, co skript dělá. Live skripty podporují interaktivní prvky, jako jsou posuvníky a tlačítka, a dají se exportovat do formátů PDF nebo HTML [4].

1.8 Řídicí struktury

Řídicí struktury v MATLABu slouží k řízení programu, tedy určují, které části kódu se vykonají a zda se některé části budou opakovat.

1.8.1 Podmínka if/else

Podmínka if/else se používá pro podmíněné vykonávání kódu na základě splnění logických podmínek. Příkaz if vyhodnotí logický výraz a pokud je pravdivý, vykoná odpovídající blok příkazů. Pokud podmínka není splněna, lze použít elseif pro testování jiných podmínek. Pokud žádná podmínka neplatí, provede se kód v bloku else. Každý podmíněný blok musí být ukončen příkazem end [3].

Obecná syntaxe kódu:

```
if podmínka
    % Kód vykonaný při splnění podmínky
elseif jiná_podmínka
    % Kód vykonaný, pokud byla splněna jiná podmínka
else
    % Kód vykonaný, pokud žádná z předchozích podmínek nebyla splněna
end
```

1.8.2 Switch

Switch se používá pro vícenásobné větvení programu na základě hodnoty výrazu. Po příkazu *switch* následuje testovaná hodnota, která se porovnává s jednotlivými možnostmi příkazu *case*. Pokud dojde ke shodě, vykoná se odpovídající blok kódu. Pokud žádná z možností neodpovídá testované hodnotě, provede se kód ve větvi *otherwise*. Stejně jako *if* musí být switch ukončen příkazem *end*. Tento přístup je přehlednější než více if/else bloků [3].

Obecná syntaxe kódu:

```
switch výraz
  case hodnota1
    % Kód, který se provede, pokud výraz odpovídá hodnotě 1
  case hodnota2
    % Kód pro případ hodnoty 2
  case hodnota3
    % Kód, pokud výraz odpovídá hodnotě 3
  otherwise
    % Kód, který se provede, pokud žádný z předchozích případů
    %nebyl splněn
end
```

1.8.3 Cyklus for

Cyklus *for* se používá v případech, kdy předem známe počet opakování. Při každém průchodu cyklu se proměnné přiřadí další hodnota ze zadaného vektoru [3].

Obecná syntaxe kódu:

```
for proměnná = začátek:krok:konec
  % Blok příkazů, který se provede v každé iteraci
end
```

1.8.4 Cyklus while

Cyklus *while* se používá v situacích, kdy není předem znám počet opakování. Opakování pokračuje, dokud je splněna podmínka uvedená za klíčovým slovem *while*. Jakmile podmínka přestane platit, cyklus se ukončí [3].

Obecná syntaxe kódu:

```
while podmínka
  % Blok příkazů, které se vykonávají při splnění podmínky
end
```

1.9 Základní příkazy

Zde je přehled základních příkazů pro práci s MATLABem [3].

1.9.1 Správa prostředí a proměnných

- **clc** – vymaže příkazové okno
- **clear** – odstraní proměnné z Workspace
- **who** – zobrazí všechny proměnné uložené v paměti
- **whos** – zobrazí seznam proměnných s podrobnými vlastnostmi
- **save** – uloží proměnné z Workspace do souboru
- **load** – načte proměnné ze souboru do Workspace
- **pwd** – zobrazí aktuální adresář
- **dir** – zobrazí obsah aktuálního adresáře
- **cd** – změni aktuálně používaný adresář
- **help** – zobrazení stručné nápovědy o funkcích a příkazech přímo v příkazovém řádku.

1.9.2 Práce s maticemi a vektory

- **[1 2 3 4]** – řádkový vektor
- **[1; 2; 3; 4]** – sloupcový vektor
- **[1 2; 3 4]** – matice 2x2
- **zeros(n,m)** – vytvoří matici nul $n \times m$
- **eye(n)** – vytvoří jednotkovou matici $n \times n$
- **rand(n,m)** – vytvoří matici $n \times m$ s náhodnými reálnými čísly od 0 do 1
- **ones(n,m)** – vytvoří matici nul $n \times m$
- **inv** – výpočet inverzní matice
- **det** – výpočet determinantu matice

1.9.3 Práce s čísly

- **abs** – absolutní hodnota
- **round** – zaokrouhlí na celé číslo
- **mod** – zbytek po dělení
- **sqrt** – druhá odmocnina
- **log** – přirozený algoritmus
- **log10** – desítkový algoritmus
- **sin** – sinus
- **cos** – cosinus

1.9.4 Práce s řetězci

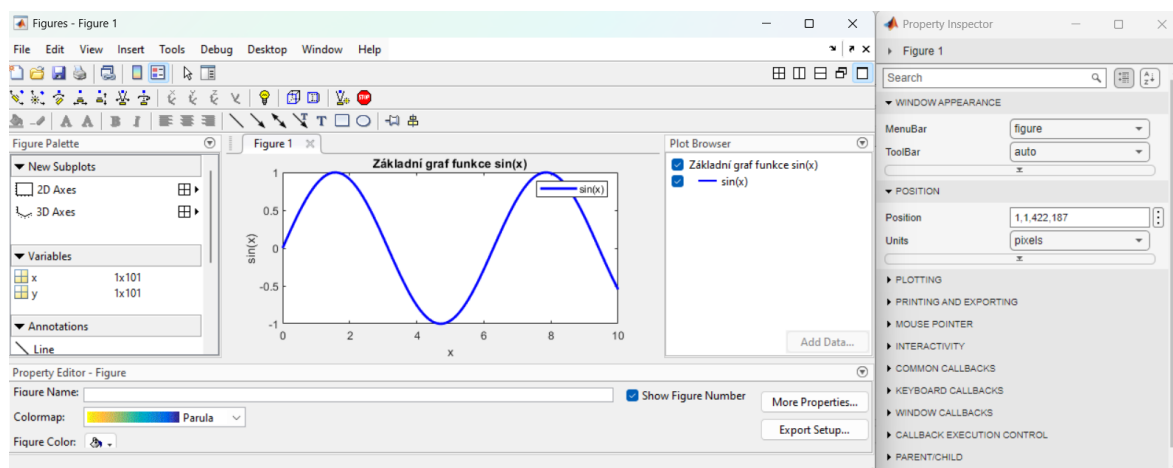
- **length** – délka řetězce
- **lower** – převod znaků na malá písmena
- **upper** – převod znaků na velká písmena
- **str2num** – převod textu na číslo
- **num2str** – převod čísla na text
- **strcat** – spojení textových řetězců
- **strrep** – nahrazení části/í řetězce za jiný řetězec/jiné řetězce

1.9.5 Práce s grafy

- **plot(x,y)** – 2D graf
- **scatter(x,y)** – bodový graf
- **plot3(x,y,z)** – 3D graf
- **hold on** – umožní vykreslit více průběhů do jednoho okna
- **legend** – přidání legendy
- **title** – název grafu
- **xlabel** – popis osy x
- **ylabel** – popis osy y
- **grid on** – zobrazí mřížku grafu

1.10 Interaktivní editace grafů

MATLAB poskytuje nástroje pro úpravy grafů prostřednictvím uživatelského rozhraní, které umožňuje uživatelům snadno měnit vlastnosti grafů, jako jsou typy grafů, barvy, styly čar, osy, popisky a legendy. Tímto způsobem lze přizpůsobit vzhled grafu bez nutnosti psát kód. Takto upravený graf lze poté převést zpět na kód [9]. Ukázka tohoto rozhraní je zobrazena na obrázku 2.



Obrázek 2 Interaktivní editace grafu

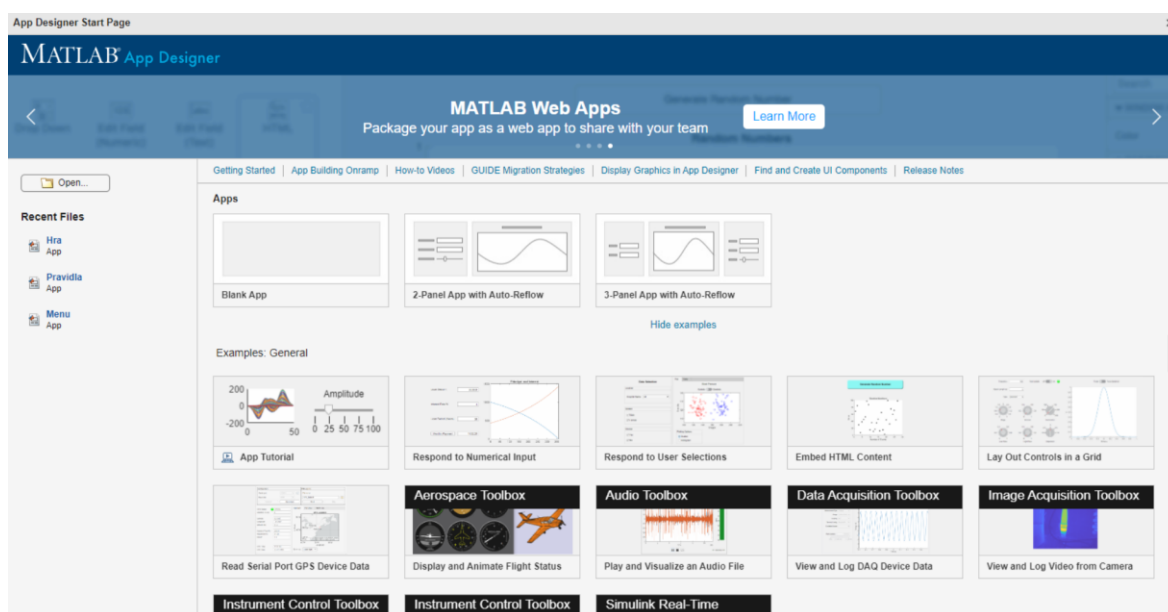
1.11 App Designer

App Designer je integrované vývojové prostředí, které umožňuje vytvářet aplikace s grafickým uživatelským rozhraním (GUI) v MATLABu. Tento nástroj poskytuje snadno použitelný vizuální designér, který umožňuje vývojářům vytvářet aplikace s tlačítky, textovými poli, grafy a dalšími ovládacími prvky. App Designer kombinuje grafický návrh s programováním, což usnadňuje vývoj aplikací uživatelům, kteří nemají velké zkušenosti s programováním.

App Designer byl do MATLABu přidán v roce 2016 jako moderní nástroj pro návrh GUI a postupně začal nahrazovat starší prostředí GUIDE [10].

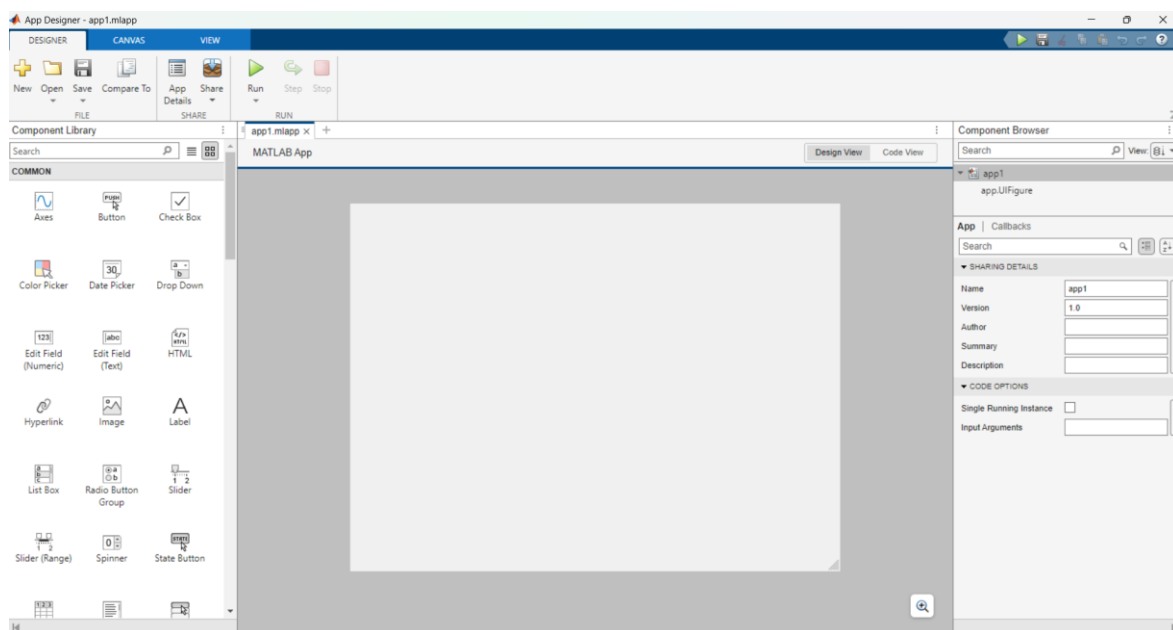
1.11.1 Prostředí App Designeru

Po spuštění App Designeru se nejprve zobrazí úvodní okno, které umožňuje uživateli zvolit mezi vytvořením nové aplikace nebo otevřením existujícího projektu. Kromě těchto možností se zde nachází také ukázkové příklady aplikací, které demonstrují různé funkce a možnosti tohoto nástroje. Uživatel si může vybrat prázdnou aplikaci a začít s vlastním návrhem, nebo využít předpřipravené šablony, které obsahují základní rozložení prvků. Díky názorným příkladům si lze rychle osvojit práci s App Designerem a inspirovat se při tvorbě vlastních aplikací. Ukázka úvodního okna je zobrazena na obrázku 3.



Obrázek 3 Úvodní okno App Designeru

Po otevření prázdné aplikace se zobrazí hlavní vývojové prostředí, které lze vidět na obrázku 4.



Obrázek 4 Vývojové prostředí App Designeru

Toto prostředí je rozděleno do několika klíčových částí:

Design View – pracovní plocha, kde lze metodou drag-and-drop přidávat a uspořádat prvky uživatelského rozhraní, jako jsou tlačítka, textová pole, posuvníky atd.

Component Library – obsahuje seznam dostupných komponentů, které lze přetáhnout do pracovní plochy. Najdeme zde prvky jako tlačítka, zaškrťovací políčka, přepínače, posuvníky atd.

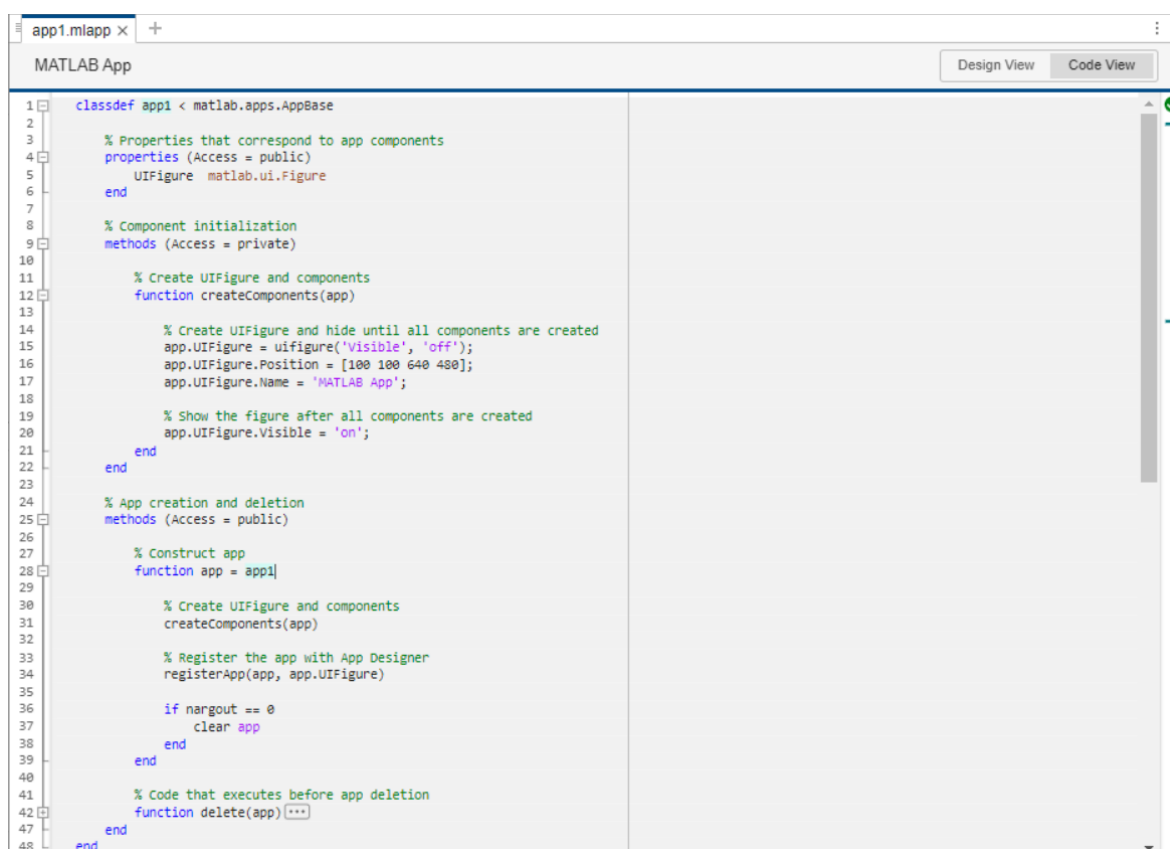
Component Browser – umožňuje nastavovat vlastnosti vybraných prvků, jako jsou popisky, barvy, velikost nebo výchozí hodnoty.

Code View – slouží k programování aplikace. MATLAB automaticky vygeneruje základní strukturu kódu, která obsahuje hlavní inicializační funkci aplikace.

1.11.2 Programování

Aplikace vytvořená v App Designeru je definována jako třída, což znamená, že všechny prvky aplikace jsou vnímány jako objektově orientované entity se svými vlastnostmi a metodami. Po navržení grafického rozhraní v *Design View* se můžeme přepnout do *Code View*, kde je automaticky vygenerován předpřipravený kód, který odpovídá námi navrženému rozhraní. Tento kód obsahuje definici třídy aplikace, inicializaci všech komponent a jejich propojení s objektem.

V *Code View* pracujeme s kódem, který nejen definuje vzhled, ale především chování aplikace. Každý prvek uživatelského rozhraní má v kódu svou odpovídající vlastnost, kterou lze dále programovat. Zásadní součástí programování jsou tzv. callback funkce, které určují reakce aplikace na uživatelské akce. Například při kliknutí na tlačítko může callback funkce spustit výpočet, nebo provést jinou operaci. Kromě definování callback funkcí lze vytvářet i funkce vlastní [10]. Ukázku z prostředí Code View lze vidět na obrázku 5.



Obrázek 5 Code View v App Designeru

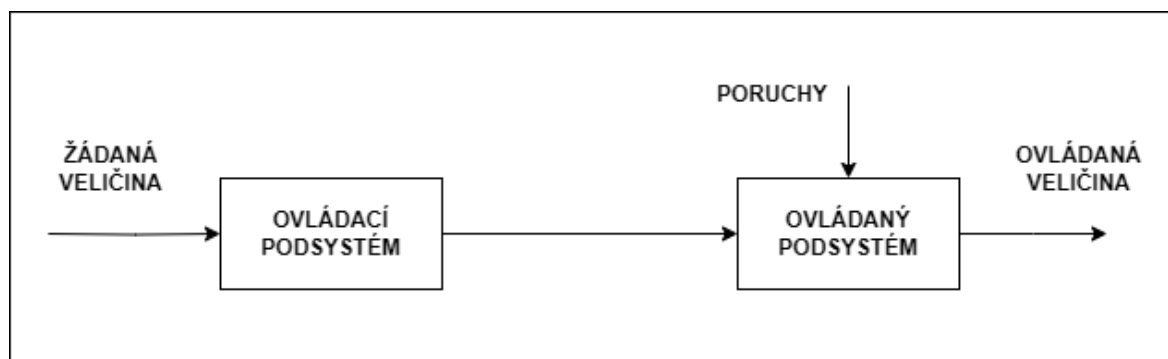
2 ZÁKLADY SPOJITÉHO ŘÍZENÍ

Řízení v oblasti automatizace představuje klíčový proces, jehož cílem je ovlivňování chování určitého systému tak, aby jeho činnost odpovídala předem stanoveným požadavkům.

2.1 Ovládání a regulace

Řízení lze rozdělit na ovládání a regulaci.

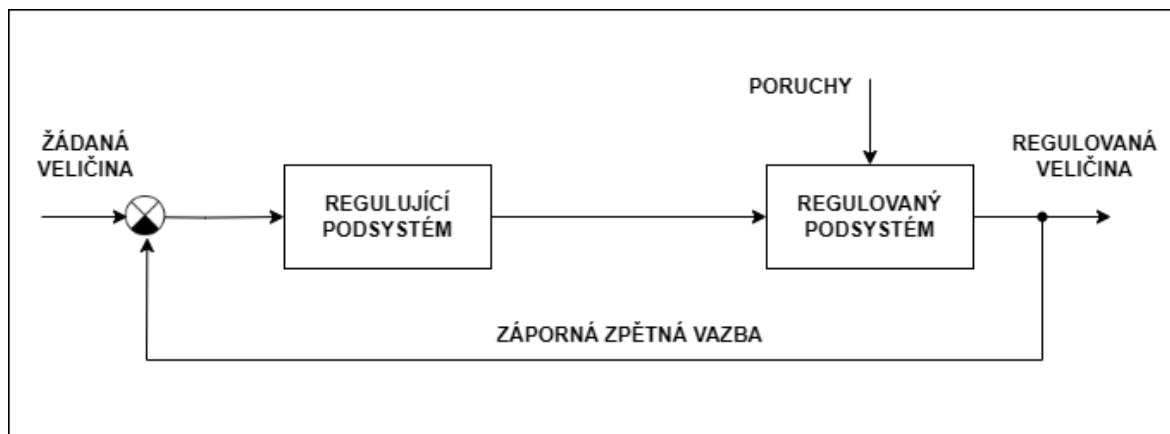
Ovládání probíhá bez zpětné vazby, což znamená, že systém nemá informace o tom, jak se změny projeví na jeho výstupu. Blokové schéma tohoto systému je znázorněno na obrázku 6. Ovládání může být manuální, kdy člověk přímo zasahuje do jeho činnosti, nebo automatické, kdy systém pracuje samostatně na základě předem nastavených parametrů.



Obrázek 6 Blokové schéma ovládání

(zdroj: vlastní zpracování podle [11])

Regulace naopak využívá zpětnou vazbu, díky níž průběžně porovnává skutečné hodnoty s požadovanými a podle rozdílu mezi nimi provádí korekce tak, aby byla odchylka od požadované hodnoty co nejmenší [11]. Blokové schéma regulace lze vidět na obrázku 7.

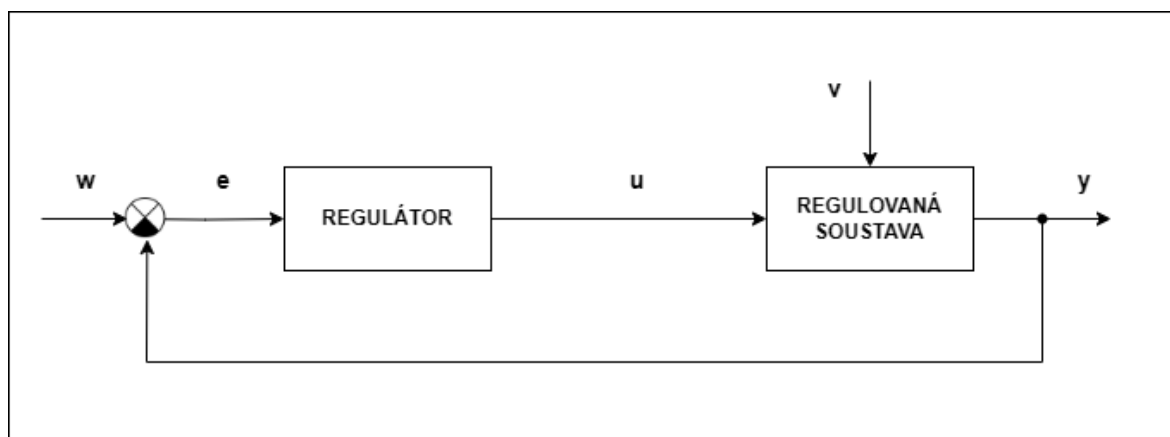


Obrázek 7 Blokové schéma regulace

(zdroj: vlastní zpracování podle [11])

2.2 Regulační obvod

Regulační obvod je systém, jehož úkolem je automaticky udržovat určitou veličinu na požadované hodnotě. Obecné schéma regulačního obvodu je znázorněno na obrázku 8. Klíčovým principem tohoto obvodu je zpětná vazba, která umožňuje neustálé sledování regulované veličiny a její přizpůsobování cílovému stavu. Pokud se naměřená hodnota odchýlí od požadované, regulátor vyšle korekční signál, který ovlivní řízený proces tak, aby se veličina co nejrychleji a co nejpresněji vrátila do požadovaného stavu [11].



Obrázek 8 Obecné blokové schéma regulačního obvodu

(zdroj: vlastní zpracování podle [11])

- **Žádaná veličina w** – představuje hodnotu, na kterou chceme systém nastavit.
- **Regulační odchylka e** – je rozdíl mezi žádanou veličinou w a skutečnou hodnotou regulované veličiny y .
- **Akční veličina u** – je výstupní hodnota regulátoru, která ovlivňuje regulovanou soustavu.
- **Regulovaná veličina y** – je skutečná hodnota veličiny, kterou se snažíme v regulačním procesu udržet na požadované hodnotě.
- **Poruchová veličina v** – je nežádoucí vstup do regulačního systému, který ovlivňuje regulovanou veličinu a způsobuje odchylku od požadované hodnoty.

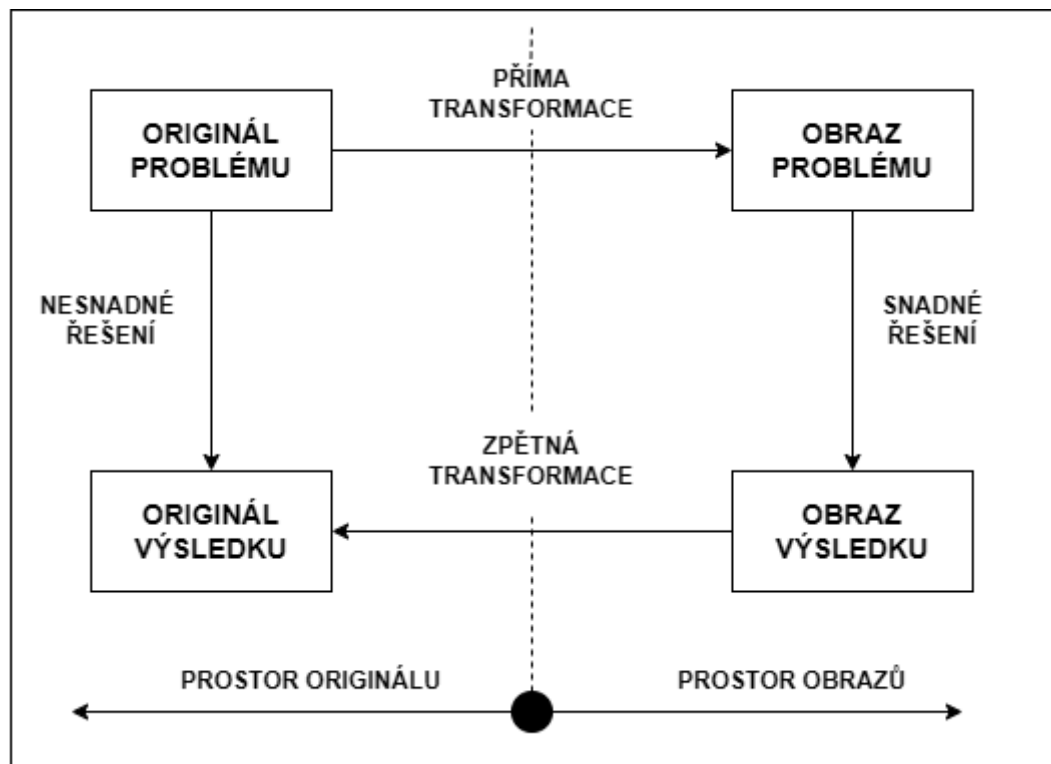
2.3 Spojité a nespojité řízení

Spojité řízení pracuje se signály, které se mění plynule v čase. Všechny vstupní i výstupní veličiny jsou spojité funkce bez skokových změn. Tento typ řízení je typický pro analogové systémy a popisuje se pomocí diferenciálních rovnic.

Nespojité (diskrétní) řízení pracuje se signály, které se mění v diskrétních časových okamžicích, obvykle v pravidelných intervalech. Vzniká tak posloupnost hodnot oddělených časovou mezerou. Tento typ řízení je typický pro digitální systémy [12].

2.4 Laplaceova transformace

Laplaceova transformace (označovaná také jako L-transformace) je matematický nástroj, který se v teorii řízení hojně využívá pro analýzu a syntézu spojitých lineárních dynamických systémů. Jejím hlavním přínosem je převod složitých problémů z časové oblasti, označované jako prostor originálů, do oblasti komplexní proměnné, tedy do tzv. prostoru obrazů. V této oblasti lze problém často řešit jednodušeji pomocí algebraických metod. Klíčovou výhodou tohoto převodu je možnost nahradit časové derivace algebraickými výrazy, což výrazně usnadňuje práci se složitými soustavami. Po nalezení řešení v obrazové oblasti se výsledek převádí zpět do časové oblasti pomocí inverzní Laplaceovy transformace [11]. Tento princip je ilustrován na obrázku 9.



Obrázek 9 Schéma řešení problému pomocí L-transformace

(zdroj: vlastní zpracování podle [11])

2.4.1 Přímá Laplaceova transformace

Přímá Laplaceova transformace $f(t)$ reálné funkce pro $t \geq 0$ je dána vztahem [11]:

$$F(s) = L\{f(t)\} = \int_0^{\infty} f(t)e^{-st} dt, \quad (1)$$

kde:

L – operátor přímé L-transformace,

$F(s)$ – obraz spojité funkce,

s – komplexní proměnná,

t – čas.

2.4.2 Zpětná Laplaceova transformace

Zpětná transformace převádí obraz $F(s)$ zpět na původní funkci [11]:

$$f(t) = L^{-1}\{F(s)\} = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)e^{st} ds. \quad (2)$$

kde:

L^{-1} – operátor zpětné L-transformace,

$F(s)$ – spojitý obraz funkce,

s – komplexní proměnná,

t – čas.

2.4.3 Podmínky použití Laplaceovy transformace

Aby bylo možné použít Laplaceovu transformaci na funkci $f(t)$, musí tato funkce splňovat následující podmínky [11]:

1. musí být nulová pro záporný čas

$$f(t) = \begin{cases} f(t) & \text{pro } t \geq 0 \\ 0 & \text{pro } t < 0 \end{cases}$$

2. musí být alespoň částečně spojitá
3. musí být exponenciálního tvaru

$$|f(t)| \leq Me^{a_0 t}$$

$$\text{kde } M > 0, a_0 \in (0, \infty), t \in (0, \infty)$$

2.4.4 Heavisideův rozvoj

Pokud má funkce obrazu $F(s)$ tvar zlomku s polynomy vyšších řádů, nelze ji obvykle přímo najít ve slovníku Laplaceovy transformace. V takovém případě je nutné ji nejprve rozložit na parciální zlomky, ke kterým pak lze nalézt originál [11].

2.5 Popis systému lineární diferenciální rovnicí

Lineární spojité systémy, které mají jeden vstup a jeden výstup, lze popsat diferenciální rovnicí s konstantními koeficienty ve tvaru [11]:

$$a_n y^n(t) + a_{n-1} y^{n-1}(t) + \dots + a_1 y'(t) + a_0 y(t) = b_m u^m(t) + \dots + b_0 u(t), \quad (3)$$

kde

a_i, b_i – konstantní koeficienty,

$u(t)$ – vstupní veličina,

$y(t)$ – výstupní veličina.

2.6 Přenos systému

Přenosová systému představuje matematický model, který popisuje vztah mezi vstupem a výstupem systému v Laplaceově oblasti. Je definována jako poměr Laplaceovy transformace výstupu ku vstupu za nulových počátečních podmínek. Obecně má tvar [11]:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + \dots + b_1 s + b_0}{a_n s^n + \dots + a_1 s + a_0}, \quad (4)$$

kde

a_i, b_i – konstantní koeficienty,

$u(t)$ – vstupní veličina,

$y(t)$ – výstupní veličina,

$U(s)$ – obraz vstupní veličiny,

$Y(s)$ – obraz výstupní veličiny.

2.7 Nuly a póly systému

Póly a nuly jsou důležitými charakteristikami přenosové funkce, které výrazně ovlivňují dynamické vlastnosti systému. Přenos systému můžeme vyjádřit ve tvaru:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_m(s-n_1) \dots (s-n_j) \dots (s-n_m)}{a_n(s-s_1) \dots (s-s_i) \dots (s-s_n)}. \quad (5)$$

Nuly systému představují kořeny čitatele přenosu. Mají vliv především na tvar a průběh výstupní odezvy.

Póly systému jsou naopak kořeny jmenovatele přenosu. Tyto póly určují dynamické vlastnosti systému a hrají klíčovou roli ve stabilitě.

Nuly i póly mohou být reálné, komplexně sdružené nebo čistě imaginární. Jejich umístění v komplexní rovině zásadně ovlivňuje chování celého systému [12].

2.8 Přechodová funkce

Přechodová funkce popisuje časovou odezvu systému na jednotkový skok za předpokladu nulových počátečních podmínek. Obvykle se značí jako $h(t)$ a obraz přechodové funkce je ve tvaru [13]:

$$h(t) = L^{-1} \left\{ \frac{G(s)}{s} \right\} \quad (6)$$

2.9 Impulzní funkce

Impulzní funkce, značená $g(t)$, představuje odezvu systému na jednotkový impulz za předpokladu nulových počátečních podmínek. Obraz impulzní funkce je dán vztahem [13]:

$$g(t) = L^{-1} \{ G(s) \} \quad (7)$$

2.10 Frekvenční přenos

Frekvenční přenos popisuje odezvu lineárního systému na harmonický signál v ustáleném stavu. Určuje, jak systém mění amplitudu a fázi sinusového vstupu v závislosti na frekvenci. Frekvenční přenos lze zapsat jako [13]:

$$G(j\omega) = G(s)_{s=j\omega} = \frac{Y(j\omega)}{U(j\omega)} = A(\omega)e^{j\varphi(\omega)}, \quad (8)$$

kde

$Y(j\omega), U(j\omega)$ – Fourierovy obrazy vstupních a výstupních signálů,

$A(\omega)$ – amplituda (zesílení),

$\varphi(\omega)$ – fázový posun.

2.11 Amplitudo-fázová frekvenční charakteristika v komplexní rovině

Amplitudo-fázová charakteristika znázorňuje, jak se systém chová při různých frekvencích. Jedná se o grafické zobrazení frekvenčního přenosu v komplexní rovině, kde každému kmitočtu odpovídá jeden bod. Spojením těchto bodů vzniká křivka, která ukazuje změnu amplitudy a fáze výstupu vzhledem ke vstupu v závislosti na frekvenci. Frekvenční přenos lze zapsat ve tvaru [12]:

$$G(j\omega) = P(\omega) + jQ(\omega) \quad (9)$$

nebo se upraví na exponenciální tvar

$$G(j\omega) = A(\omega)e^{j\varphi(\omega)}, \quad (10)$$

kde modul přenosu je ve tvaru:

$$A(\omega) = \sqrt{P^2(\omega) + Q^2(\omega)} \quad (11)$$

a fáze ve tvaru:

$$\varphi(\omega) = \arctan\left(\frac{Q(\omega)}{P(\omega)}\right). \quad (12)$$

2.12 Frekvenční charakteristika v logaritmických souřadnicích

Pro lepší přehlednost a jednodušší analýzu se frekvenční přenos často zobrazuje v logaritmických souřadnicích. Přenos je nejprve vyjádřen v exponenciálním tvaru [12]:

$$G(j\omega) = A(\omega)e^{j\varphi(\omega)}. \quad (13)$$

Logaritmická charakteristika se pak skládá ze dvou složek:

1. Logaritmická amplitudová charakteristika

$$A(\omega)[dB] = |G(j\omega)|_{dB} = 20 \log_{10} |G(j\omega)| \quad (14)$$

2. Logaritmická fázová charakteristika

$$\varphi(\omega) = \arg G(j\omega) = \arctan \left(\frac{Q(\omega)}{P(\omega)} \right) \quad (15)$$

2.13 Regulátor

Regulátor je prvek v řídicím systému, který vyhodnocuje odchylku mezi požadovanou a skutečnou hodnotou regulované veličiny a na základě toho generuje korekční zásah. Jeho cílem je minimalizovat tuto odchylku a zajistit stabilní a přesné řízení systému.

2.13.1 Členění regulátorů

Regulátory lze rozdělit dle několika kritérií, která určují jejich vlastnosti a způsob fungování [12].

1. Podle typu přenášeného signálu

- **Spojitý regulátor** – vstupní i výstupní signál je spojitou funkcí času, což znamená, že může plynule nabývat jakýchkoliv hodnot v každém časovém okamžiku.
- **Nespojitý (diskrétní) regulátor** – vstup, výstup nebo oba signály se mění s časem nespojitě. To znamená, že v určitých časových okamžicích dochází ke skokové změně z jedné hodnoty na jinou, která zůstává konstantní až do další skokové změny.

2. Podle zdroje energie

- **Přímé regulátory** – nepotřebují vlastní zdroj energie, veškerou energii potřebnou ke své činnosti získávají přímo z regulované soustavy.
- **Nepřímé regulátory** – odebírají energii z napájecího zdroje.

3. Podle nositele signálu

- **Pneumatické regulátory** – signál je přenášen pomocí tlaku stlačeného vzduchu.
- **Hydraulické regulátory** – signál je přenášen pomocí tlaku kapaliny (nejčastěji oleje).
- **Elektrické regulátory** – signál je přenášen prostřednictvím elektrické energie

2.13.2 Spojité regulátory

Jak již bylo řečeno, spojitý regulátor je takový regulátor, u něhož výstupní veličina plynule reaguje na změny vstupní veličiny a může nabývat libovolných hodnot v celém rozsahu. Všechny prvky regulačního obvodu pracují spojitě, což znamená, že regulovaná veličina neustále ovlivňuje akční veličinu, která se mění plynule bez skokových změn. Díky tomu spojité regulátory dosahují vyšší přesnosti, ačkoliv jejich zapojení bývá složitější a jejich nastavení náročnější než u nespojitých regulátorů. Přesto jsou preferovanou volbou tam, kde je vyžadována precizní regulace a stabilní udržení požadované hodnoty [12].

2.13.3 Realizace spojitých regulátorů

Spojité regulátory jsou založeny na třech základních složkách: proporcionální (P), integrační (I) a derivační (D), které lze vzájemně kombinovat a vytvářet tak složitější systémy, jako jsou PI, PD a PID regulátory [12].

Základní složky:

- **Proporcionální složka (P)** – tato složka funguje jako zesilovač. Akční veličina se mění úměrně velikosti regulační odchylky. Čím větší je regulační odchylka, tím silnější je zásah regulátoru.
- **Integrační složka (I)** – akční veličina je úměrná integrálu regulační odchylky. Reaguje na regulační odchylku v čase a postupně přizpůsobuje výstup tak, aby byla regulační odchylka trvale odstraněna.
- **Derivační složka (D)** – akční veličina je úměrná derivaci regulační odchylky. Reaguje na rychlost změny regulační odchylky v čase. Upravuje regulační zásah tak, aby byly zmírněny překmity a zrychlila se reakce systému.

Jednotlivé složky se mohou používat samostatně kromě složky D, u které samostatné použití nemá význam.

Spojením všech tří složek vznikne proporcionálně integračně derivační regulátor (zkráceně PID regulátor). Jeho vlastnosti lze popsat rovnicí [12]:

$$u(t) = \underbrace{r_0 e(t)}_P + \underbrace{r_{-1} \int_0^t e(\tau) d\tau}_I + \underbrace{r_1 \frac{de(t)}{dt}}_D = k_p \left[e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau + T_D \frac{de(t)}{dt} \right] \quad (16)$$

kde

r_0 – váha proporcionální složky regulátoru,

r_{-1} – váha integrační složky regulátoru,

r_1 – váha derivační složky regulátoru,

T_I – integrační časová konstanta regulátoru,

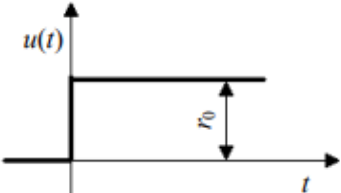
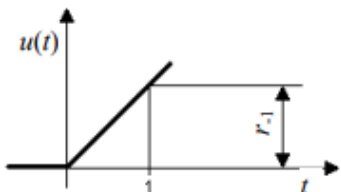
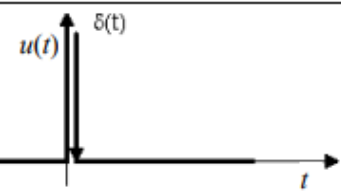
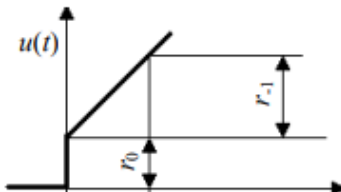
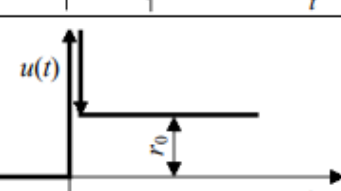
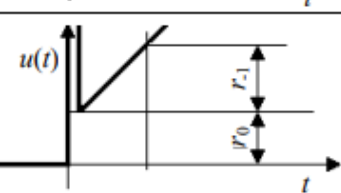
T_D – derivační časová konstanta regulátoru,

$k_p = r_0$ – zesílení regulátoru.

Po použití Laplaceovy transformace za předpokladu nulových počátečních podmínek získáme přenos ideálního PID regulátoru

$$G_R(s) = \frac{U(s)}{E(s)} = r_0 + \frac{r_{-1}}{s} + r_1 s = k_p \left(1 + \frac{1}{T_I s} + T_D s \right). \quad (17)$$

Přehled běžně používaných regulátorů je uveden na obrázku 10.

Typ	Rovnice	Přechodová charakteristika	Přenos $G_R(s)$
P	$u(t) = r_0 e(t)$		$G_R(s) = r_0$
I	$u(t) = r_{-1} \int_0^t e(\tau) d\tau$		$G_R(s) = \frac{r_{-1}}{s}$
D - člen	$u(t) = r_1 \frac{de(t)}{dt}$		$G_R(s) = r_1 s$
PI	$u(t) = r_0 e(t) + r_{-1} \int_0^t e(\tau) d\tau$		$G_R(s) = r_0 + \frac{r_{-1}}{s}$
PD	$u(t) = r_0 e(t) + r_1 \frac{de(t)}{dt}$		$G_R(s) = r_0 + r_1 s$
PID	$u(t) = r_0 e + r_{-1} \int_0^t e(\tau) d(\tau) + r_1 \frac{de(t)}{dt}$		$G_R = r_0 + \frac{r_{-1}}{s} + r_1 s$

Obrázek 10 Přehled a vlastnosti ideálních regulátorů

(zdroj: [14])

2.14 Regulovaná soustava

Regulovaná soustava je systém, na kterém se provádí regulace, tedy udržování regulované veličiny na požadované hodnotě. Regulovaná soustava přijímá vstupní akční veličinu a na jejím základě mění svou výstupní regulovanou veličinu.

Vlastnosti regulované soustavy zásadně ovlivňují průběh regulačního procesu, a proto je pro úspěšný návrh regulace nutná znalost dynamických vlastností soustavy.

Dynamické vlastnosti zahrnují chování systému v čase (přechodové charakteristiky, rychlost odezvy, setrvačnost) [14].

2.15 Typy regulovaných soustav

2.15.1 Proporcionální (statické) regulované soustavy

Tyto soustavy mají schopnost se samovolně ustálit na nové rovnovážné hodnotě. Neboli po skokové změně akční veličiny se výstupní veličina samovolně ustálí na nové rovnovážné hodnotě bez působení regulátoru. Dynamické vlastnosti proporcionální regulované soustavy lze vyjádřit obecnou lineární rovnicí n -tého řádu [14]:

$$a_n y^n(t) + a_{n-1} y^{n-1}(t) + \dots + a_0 y(t) = b_0 u(t). \quad (18)$$

Řád diferenciální rovnice odpovídá řádu soustavy.

1. Soustava 0. řádu

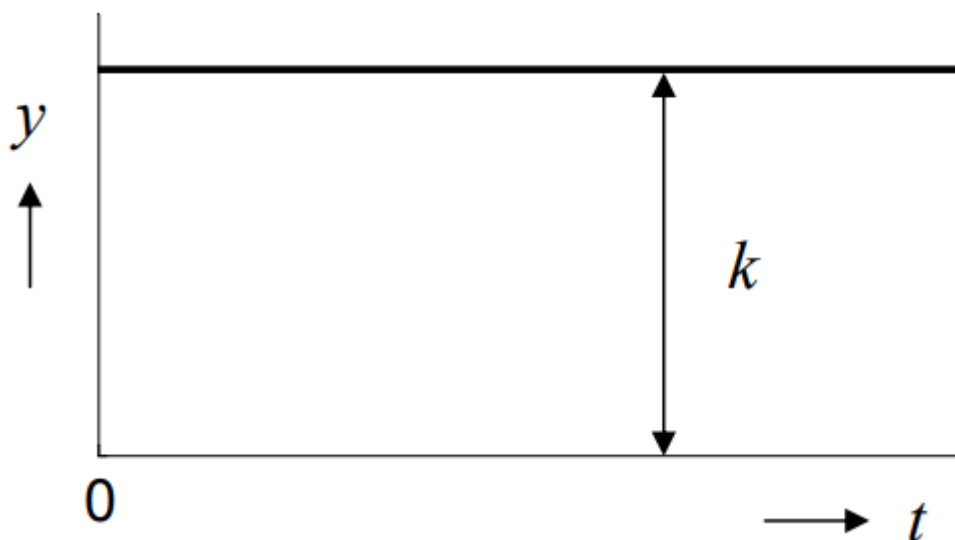
Diferenciální rovnice:

$$a_0 y(t) = b_0 u(t) \quad (19)$$

Přenos soustavy:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{a_0} = k, \quad (20)$$

kde k vyjadřuje zesílení soustavy. Na obrázku 11 je zobrazena přechodová charakteristika proporcionální soustavy 0. řádu.



Obrázek 11 Přejchodová charakteristika proporcionalní soustavy 0. řádu

(zdroj: [14])

2. Soustava 1. řádu

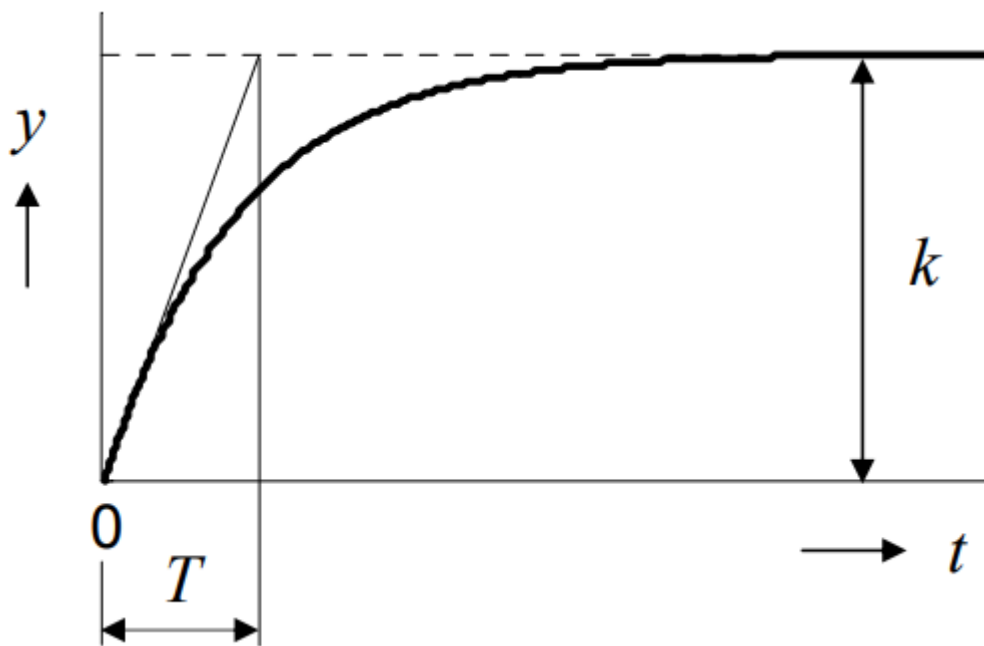
Diferenciální rovnice:

$$a_1 y'(t) + a_0 y(t) = b_0 u(t) \quad (21)$$

Přenos soustavy:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{a_1 s + a_0} = \frac{k}{Ts + 1}, \quad (22)$$

kde T je časová konstanta. Přejchodová charakteristika soustavy 1. řádu je zobrazena na obrázku 12.



Obrázek 12 Přejchodová charakteristika proporcionální soustavy 1. řádu

(zdroj: [14])

3. Soustava 2. řádu

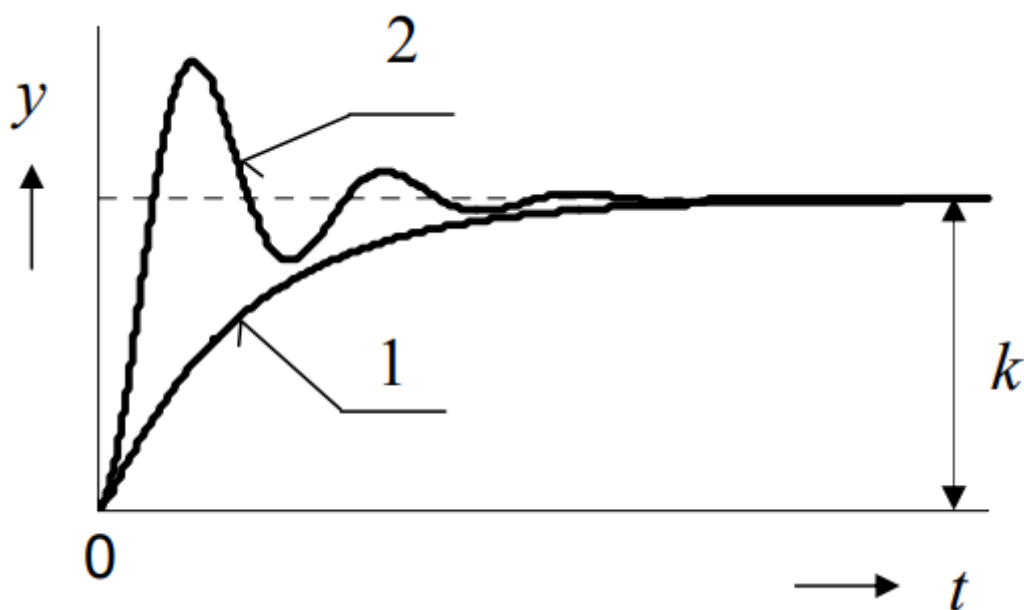
Diferenciální rovnice:

$$a_2 y''(t) + a_1 y'(t) + a_0 y(t) = b_0 u(t) \quad (23)$$

Přenos soustavy:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{a_2 s^2 + a_1 s + a_0} = \frac{k}{(T_1 s + 1) + (T_2 s + 1)}, \quad (24)$$

kde T je časová konstanta. Obrázek 13 znázorňuje přechodovou charakteristiku proporcionální soustavy 2. řádu.



Obrázek 13 Přechodová charakteristika proporcionální soustavy 2. řádu

(zdroj: [14])

2.15.2 Integrační (astatické) regulované soustavy

Integrační regulované soustavy nemají samoregulační schopnost, na rozdíl od soustav proporcionálních. Po vyvedení soustavy z rovnovážného stavu nedojde k ustálení bez zásahu regulátoru. Výstupní veličina se po odeznění přechodového děje mění konstantní rychlostí [14].

1. Integrační soustava 0. řádu

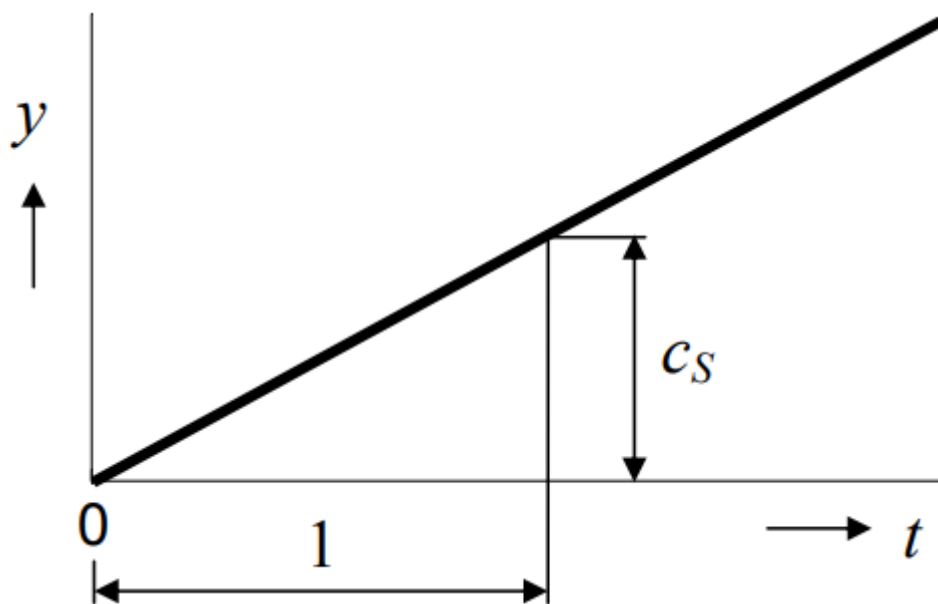
Diferenciální rovnice:

$$a_1 y'(t) = b_0 u(t) \quad (25)$$

Přenos soustavy:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{a_0} \frac{1}{s} = \frac{c_s}{s} \quad (26)$$

Přechodová charakteristika integrační soustavy 0. řádu je ilustrována na obrázku 14.



Obrázek 14 Přejchodová charakteristika integrační soustavy 0. řádu

(zdroj: [14])

2. Integrační soustava 1. řádu

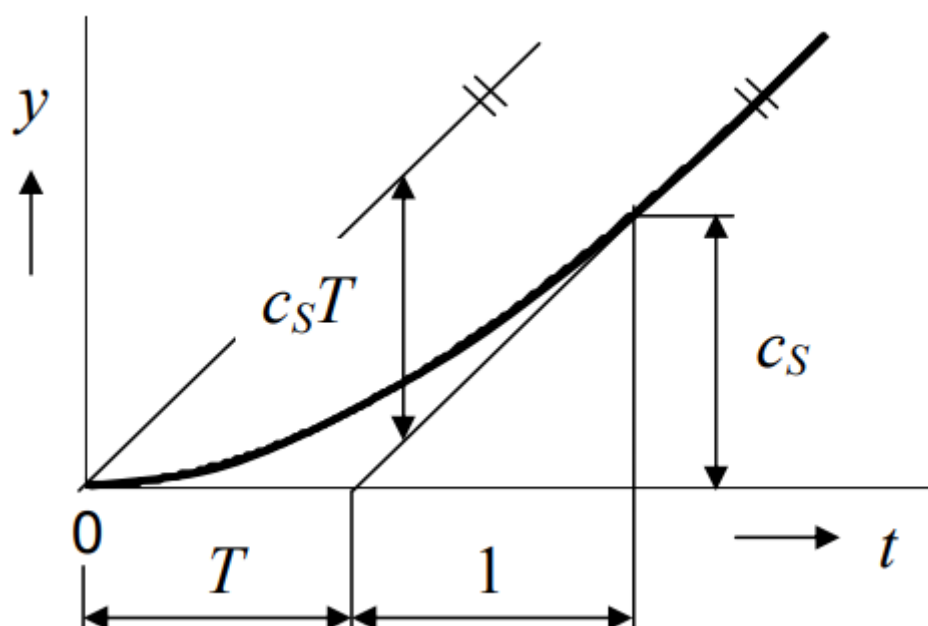
Diferenciální rovnice:

$$a_2 y''(t) + a_1 y'(t) = b_0 u(t) \quad (27)$$

Přenos soustavy:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{s(a_2 s + a_1)} = \frac{c_s}{s(Ts + 1)} \quad (28)$$

Na obrázku 15 je znázorněna přechodová charakteristika integrační soustavy 1. řádu.



Obrázek 15 Přechodová charakteristika integrační soustavy 1. řádu

(zdroj: [14])

2.15.3 Regulované soustavy s dopravním zpožděním

Dopravní zpoždění představuje časovou prodlevu mezi změnou vstupní veličiny a odpovídající reakcí výstupní veličiny. Toto zpoždění vzniká v důsledku doby potřebné k přenosu informace, energie nebo látky v systému [14].

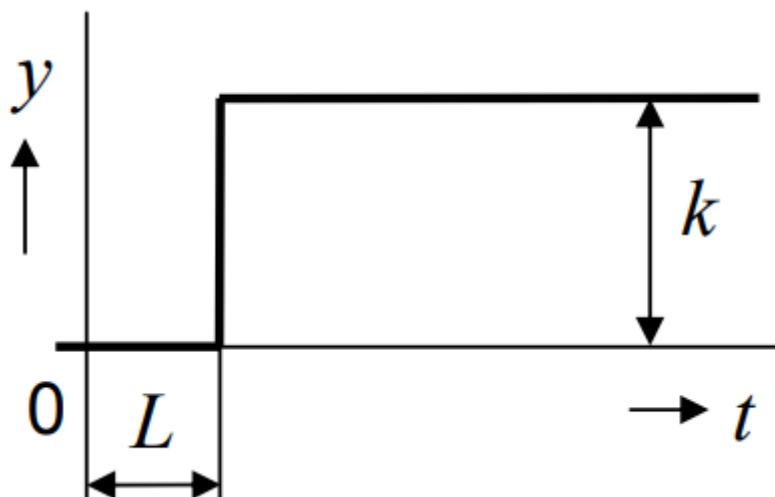
Diferenciální rovnice:

$$a_0 y(t) = b_0 u(t - L) \quad (29)$$

Přenos soustavy:

$$G_s(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{a_0} e^{-Ls} = k e^{-Ls}, \quad (30)$$

kde T_d představuje dopravní zpoždění. Přechodová charakteristika soustavy s dopravním zpožděním je zobrazena na obrázku 16.



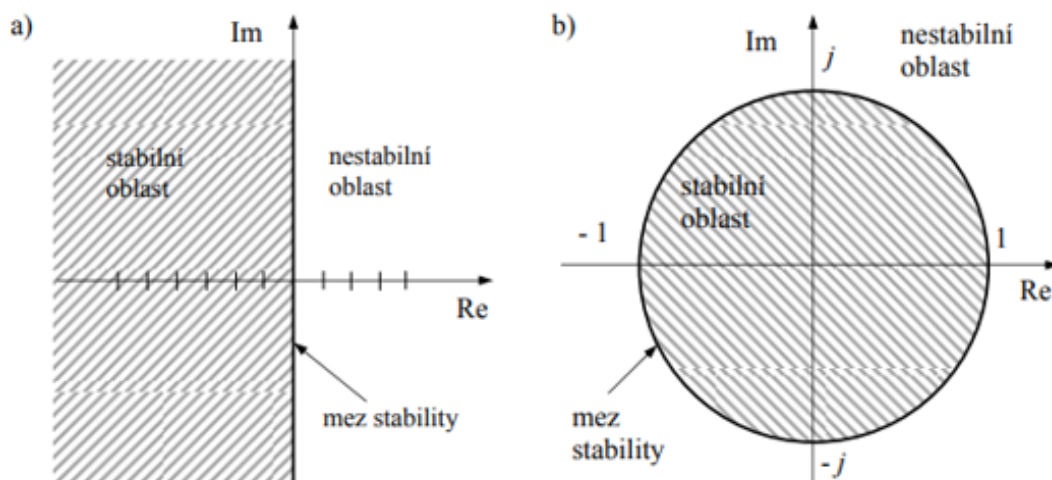
Obrázek 16 Přechodová charakteristika regulované soustavy s dopravním zpožděním

(zdroj: [14])

2.16 Stabilita systému

Stabilita systému je jeho schopnost navrátit se do původního nebo jiného rovnovážného stavu po vychýlení způsobeném poruchou či změnou požadované veličiny.

Spojitý systém je stabilní, pokud všechny póly systému (kořeny jeho charakteristické rovnice) leží v levé polorovině komplexní roviny. To znamená, že všechny koeficienty charakteristického polynomu mají stejné znaménko a žádný z nich není nulový. Naopak diskrétní systém je stabilní, pokud všechny póly systému leží uvnitř jednotkové kružnice v rovině. Obrázek 17 znázorňuje stabilitu systémů v komplexní rovině pro spojitý (a) i diskrétní (b) systémy.



Obrázek 17 Stabilita spojitých systémů (a) a diskrétních systémů (b)

(zdroj: [14])

Pro zhodnocení stability systému je nezbytné analyzovat rozmístění kořenů charakteristické rovnice a na základě jejich polohy určit, zda je systém stabilní. Pokud je nalezení kořenů komplikované, zejména u rovnic vyššího řádu, lze stabilitu posoudit prostřednictvím různých kritérií stability, která mohou být jak pro spojité, tak pro diskrétní systémy. Tato kritéria umožňují ověřit stabilitu dynamického systému bez nutnosti přímého výpočtu jeho pólů [11].

2.17 Kritéria stability

Rozlišujeme dvě základní skupiny kritérií stability, a to algebraická a geometrická. Algebraická kritéria nejsou vhodná pro analýzu stability systémů s dopravním zpožděním, jelikož pouze určují, zda je systém stabilní, aniž by poskytovala další informace. Naopak, geometrická kritéria, byť výpočetně náročnější než algebraická, umožňují nejen stanovit stabilitu systému, ale také určit míru jeho stability, včetně systémů s dopravním zpožděním [12].

2.17.1 Algebraická kritéria stability

Algebraická kritéria umožňují posoudit stabilitu bez výpočtu vlastních čísel, pouze na základě charakteristického polynomu n -tého stupně uzavřeného regulačního obvodu. Tento polynom získáme ze jmenovatele přenosové funkce [12]:

$$G_w(s) = \frac{Y(s)}{W(s)} = \frac{G_R(s) + G_S(s)}{1 + G_R(s) + G_S(s)} \quad (31)$$

kde

$G_w(s)$ – přenos řízení,

$Y(s)$ – výstupní veličina,

$W(s)$ – vstupní veličina,

$G_R(s)$ – přenos regulátoru,

$G_S(s)$ – přenos regulované soustavy.

Charakteristický polynom má tvar:

$$N(s) = a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0 \quad (32)$$

2.17.2 Routh-Schurovo kritérium stability

Toto kritérium jednoduchou redukcí kořenů určuje, zda kořeny charakteristického polynomu leží ve stabilní oblasti či nikoliv [12].

Postup použití kritéria:

1. Koeficienty charakteristického polynomu vypíšeme do řady vedle sebe od nejvyšší mocniny po nejnižší.
2. Každý sudý koeficient pro přehlednost výpočtu podtrhneme.
3. Tyto sudé koeficienty násobíme podílem prvních dvou koeficientů vynásobených hodnotou -1. Hodnoty sepíšeme pod liché koeficienty o jednu pozici vlevo.
4. Tuto novou řadu přičteme k první řadě koeficientů. Pokud jsou všechny koeficienty v druhé řadě kladné postup opakujeme. Každou iterací se nám počet koeficientů o jeden zkrátí.
5. Pokud při některé redukci vyjde záporný koeficient, můžeme skončit výpočet, jelikož můžeme usoudit, že charakteristický polynom má nestabilní kořen.
6. Postup opakujeme tak dlouho, až dostaneme řadu s posledními třemi kladnými koeficienty. V tomto bodě můžeme usoudit, že charakteristický polynom má všechny kořeny ve stabilní oblasti.

$$\begin{array}{ccccccccc}
 a_n & & \underline{a_{n-1}} & & a_{n-2} & & \underline{a_{n-3}} & & a_{n-4} & & \underline{a_{n-5}} & & \dots \\
 -\frac{a_n}{a_{n-1}}a_{n-1} & & & & -\frac{a_n}{a_{n-1}}a_{n-3} & & & & -\frac{a_n}{a_{n-1}}a_{n-5} & & & &
 \end{array} \quad (33)$$

$$\begin{array}{ccccccccccc}
 0 & a_{n-1} & a_{n-2} & -\frac{a_n}{a_{n-1}}a_{n-3} & a_{n-3} & a_{n-4} & -\frac{a_n}{a_{n-1}}a_{n-5} & a_{n-5} & \dots & & \\
 & & & \underline{\hspace{2cm}} & & & \underline{\hspace{2cm}} & & & & \\
 & & & & & & \vdots & & & & \\
 & & & & & & \vdots & & & & \\
 & & & & & & a_2 & a_1 & a_0 & > 0 &
 \end{array} \quad (34)$$

2.17.3 Hurwitzovo kritérium stability

Při určení stability pomocí Hurwitzova kritéria opět vycházíme z charakteristického polynomu. Při výpočtu se vychází z Hurwitzovy matice, v níž se počítají všechny subdeterminanty odpovídající hlavnímu determinantu matice. Systém je stabilní, pokud jsou všechny subdeterminanty větší než nula. Pokud je subdeterminant H_{n-1} roven nule, tak se systém nachází na hranici stability (systém má 2 čistě imaginární kořeny) [11].

$$H = \begin{bmatrix} a_{n-1} & a_{n-3} & a_{n-5} & \dots & 0 \\ a_n & a_{n-2} & a_{n-4} & \dots & 0 \\ 0 & a_{n-1} & a_{n-3} & \dots & 0 \\ 0 & a_n & a_{n-2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix} \quad (35)$$

$$H = \begin{bmatrix} a_{n-1} & a_{n-3} & a_{n-5} \\ a_n & a_{n-2} & a_{n-4} \\ 0 & a_{n-1} & a_{n-3} \end{bmatrix} \quad (36)$$

$$H_1 = |a_{n-1}| \quad (37)$$

$$H_2 = \begin{vmatrix} a_{n-1} & a_{n-3} \\ a_n & a_{n-2} \end{vmatrix} \quad (38)$$

$$H_3 = \begin{vmatrix} a_{n-1} & a_{n-3} & a_{n-5} \\ a_n & a_{n-2} & a_{n-4} \\ 0 & a_{n-1} & a_{n-3} \end{vmatrix} \quad (39)$$

2.17.4 Geometrická kritéria stability

Geometrická kritéria stability se využívají pro posouzení stability systémů na základě frekvenční charakteristiky obvodu. Na rozdíl od algebraických kritérií, která se zaměřují na polohu koeficientů charakteristické rovnice, geometrická kritéria analyzují chování systému v komplexní rovině při měnící se frekvenci [11].

2.17.5 Michajlovo-Leonhardovo kritérium stability

Michajlovo-Leonhardovo kritérium stability vychází z charakteristického polynomu uzavřeného regulačního obvodu:

$$N(s) = a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0, \quad (40)$$

do kterého dosadíme komplexní kmitočet za komplexní proměnnou a dostaneme tak Michajlovovu funkci:

$$\begin{aligned} N(j\omega) &= N(s)|_{s=j\omega} = a_n (j\omega)^n + a_{n-1} (j\omega)^{n-1} + \dots + a_1 (j\omega) + a_0 \\ &= U(\omega) + jV(\omega), \end{aligned} \quad (41)$$

kde

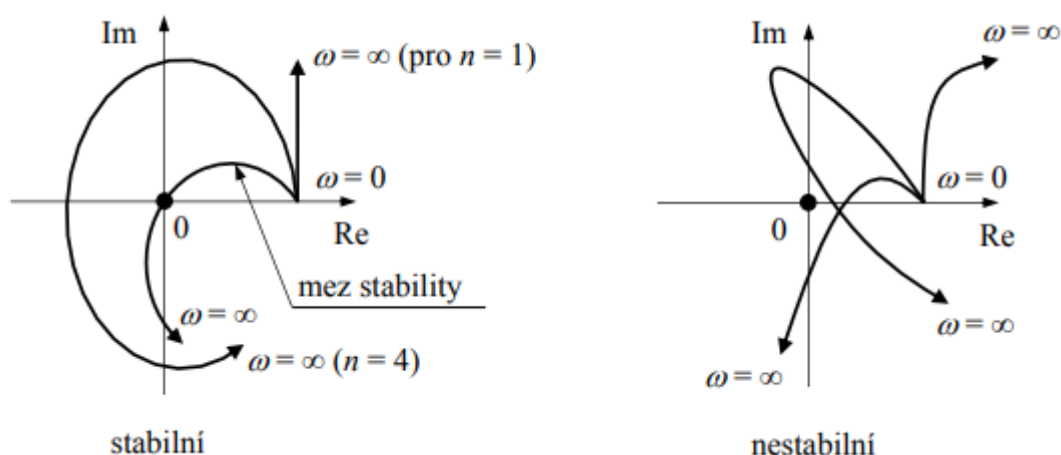
$$U(\omega) = a_0 - a_2 \omega^2 + a_4 \omega^4 - \dots \quad (42)$$

je reálná část a

$$V(\omega) = a_1 \omega - a_3 \omega^3 + a_5 \omega^5 - \dots \quad (43)$$

je část imaginární.

Systém lze považovat za stabilní, pokud Michajlovova křivka vychází z kladné reálné poloosy a postupně prochází proti směru hodinových ručiček přes n kvadrantů, přičemž jejich počet odpovídá stupni charakteristického polynomu [11]. Na obrázku 18 jsou vyobrazeny Michajlovovy křivky pro stabilní a nestabilní systém.



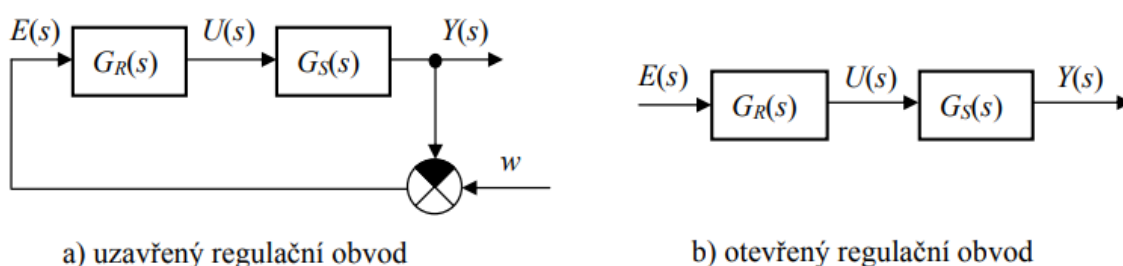
Obrázek 18 Porovnání Michajlovovy křivky pro stabilní a nestabilní systém

(zdroj: [14])

2.17.6 Nyquistovo kritérium stability

Nyquistovo kritérium stability je frekvenční metoda, která umožňuje posoudit stabilitu uzavřeného regulačního obvodu na základě kmitočtové charakteristiky otevřeného regulačního obvodu. Porovnání těchto dvou obvodů můžeme vidět na obrázku 19.

Na rozdíl od Hurwitzova či Michajlovova kritéria, která vycházejí z charakteristického polynomu, Nyquistovo kritérium pracuje s vlastnostmi otevřeného regulačního obvodu. Je zvláště vhodné pro systémy s dopravním zpožděním a může být použito i pro některé nelineární regulační obvody.



Obrázek 19 Porovnání uzavřeného a otevřeného regulačního obvodu pro aplikaci Nyquistova kritéria

(zdroj [14])

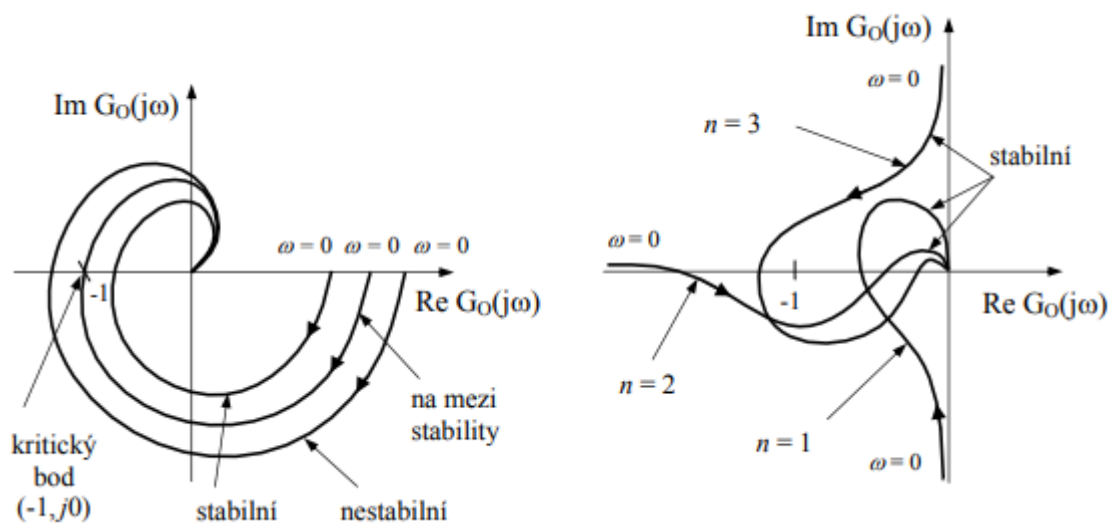
Přenos otevřeného regulačního obvodu je ve tvaru:

$$G_0(s) = G_R(s) + G_S(s) \quad (44)$$

dosazením komplexního kmitočtu za komplexní proměnnou dostaneme frekvenční přenos otevřeného regulačního obvodu.

$$G_0(s) = G_0(j\omega)|_{s=j\omega} = P(\omega) + jQ(\omega). \quad (45)$$

Dosazením frekvence ω dostaneme průběh Nyquistovy křivky, ze které určíme stabilitu uzavřeného obvodu pro stabilní otevřený regulační obvod. Pokud je otevřený obvod stabilní, uzavřený regulační obvod zůstane stabilní tehdy, když Nyquistova křivka (amplitudo-fázová frekvenční charakteristika) otevřeného obvodu neobklopuje kritický bod $[-1, j0]$ na záporné reálné poloose [11]. Průběhy Nyquistovy křivky pro různé případy stability jsou zobrazeny na obrázku 20.



Obrázek 20 Průběhy Nyquistovy křivky otevřeného regulačního obvodu v komplexní rovině

(zdroj: [14])

3 ANALÝZA EXISTUJÍCÍCH PODPOR VÝUKY TEORIE ŘÍZENÍ

V oblasti výuky teorie řízení je k dispozici řada různých studijních materiálů, které vznikly díky práci vyučujících, odborníků nebo samotných studentů. Mezi ně patří také bakalářské a diplomové práce, které se této problematice věnují.

Je však třeba brát v úvahu, že mnoho těchto prací pochází z dřívějších let, a proto některé technologie, na nichž jsou založeny, již nemusí být plně funkční nebo kompatibilní s dnešními systémy. To platí zejména pro výukové hry, webové stránky nebo softwarové aplikace, které byly vyvinuty v prostředích, jež se od té doby výrazně změnily nebo jsou dnes zastaralé. Přesto tyto práce stále představují užitečný zdroj informací.

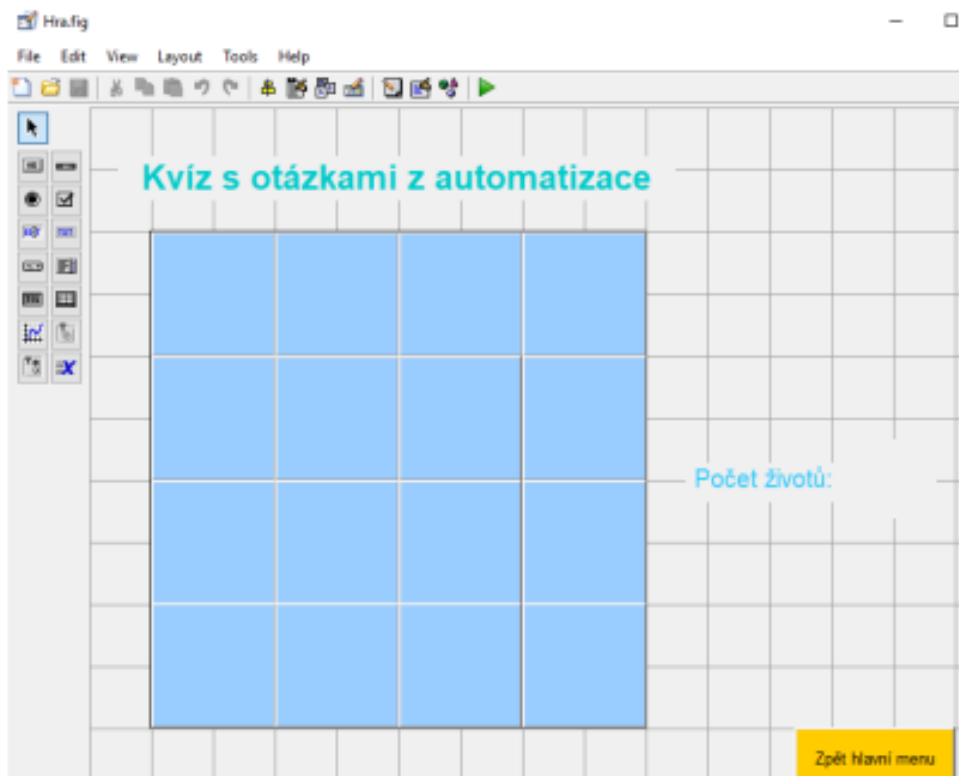
3.1 Návrh aplikace a sady příkladů pro podporu výuky automatizace pro prezenční a kombinované studium

Bakalářská práce Petry Opluštilové se zabývá návrhem aplikace určené k usnadnění výuky automatizace pro studenty prezenčního i kombinovaného studia. Aplikace je vytvořena jako interaktivní hra formou kvízu a byla vytvořena v prostředí MATLAB.

Teoretická část práce představuje prostředí MATLAB a Simulink, vysvětluje základy automatizace a podrobně se zaměřuje na lineární systémy, včetně Laplaceovy a Z-transformace.

Praktická část se soustředí na návrh a realizaci výukové hry. Podrobně popisuje její pravidla a parametry a přibližuje ji jak z pohledu programátora, tak i uživatele. Aplikace obsahuje nejen kvízové otázky ale i sadu řešených úloh z oblasti automatizace. Ukázku ze hry můžeme vidět na obrázku 21.

Tato práce nabízí užitečný nástroj pro výuku automatizace, který propojuje teoretické znalosti s interaktivní formou učení, což studentům usnadňuje pochopení dané problematiky [15].



Obrázek 21 Ukázka ze hry Kviz s otázkami z automatizace

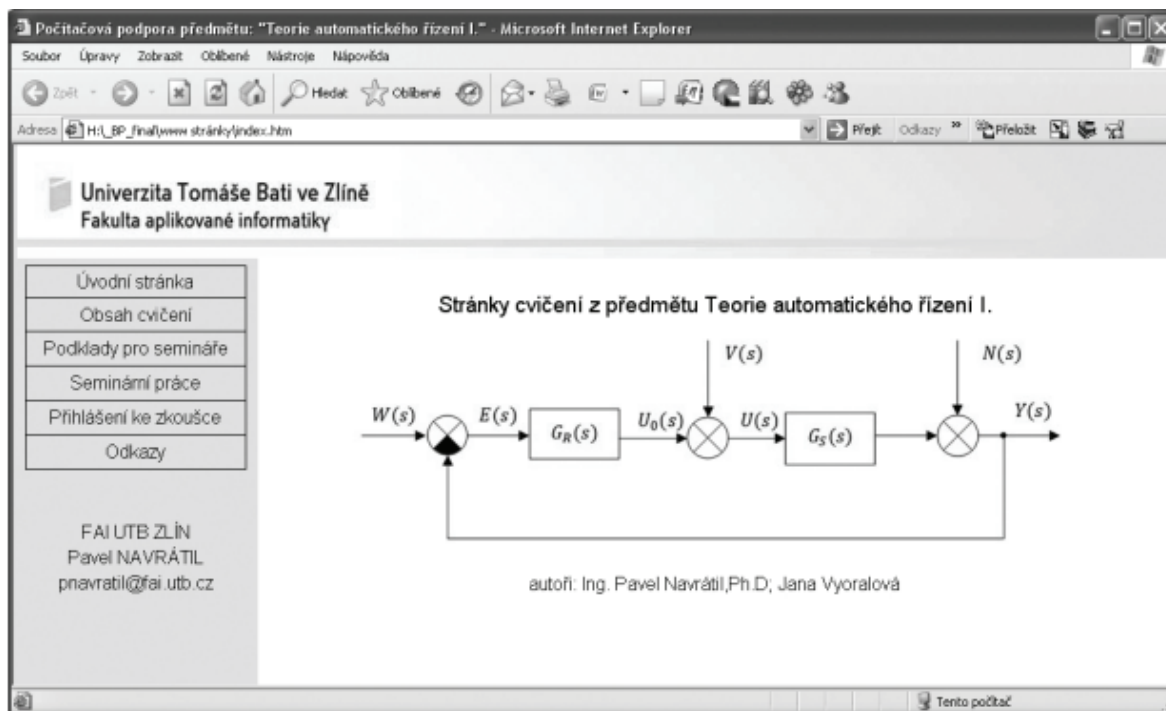
(zdroj: [15])

3.2 Podpora cvičení z předmětu: Teorie automatického řízení I.

Tato bakalářská práce Jany Vyoralové se zaměřuje na vytvoření elektronické učební pomůcky pro studenty předmětu Teorie automatického řízení I. Jejím cílem bylo zpřístupnit důležité informace nejen formou teorie, ale také pomocí praktických příkladů a simulací v prostředí MATLAB.

Součástí práce je návrh a realizace webových stránek, které slouží k publikaci těchto výukových materiálů. Ukázka webových stránek je zobrazena na obrázku 22.

V závěru autorka porovnává jednotlivé metody syntézy regulátorů na základě výsledků simulací. Práce tak přináší moderní výukovou pomůcku, která propojuje teorii s praxí a usnadňuje studentům pochopení automatického řízení [16].



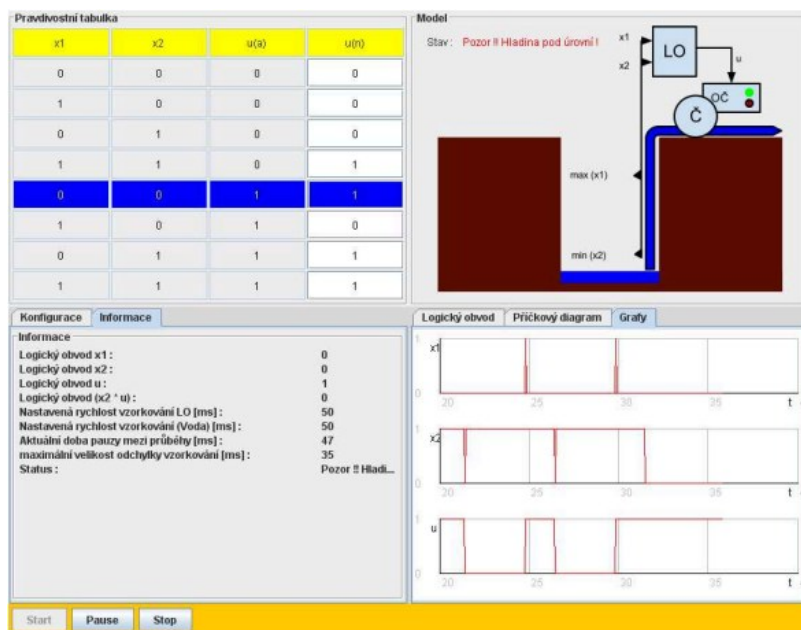
Obrázek 22 Ukázka webových stránek

(zdroj: [16])

3.3 Grafický nástroj pro podporu výuky teorie systémů a řízení

Diplomová práce Ladislava Hausekera se zaměřuje na vývoj grafického nástroje pro podporu výuky Teorie systémů a řízení. Cílem bylo vytvořit aplikace, které studentům usnadní pochopení řízení procesů a regulace. Ukázku z jedné z výukových aplikací můžeme vidět na obrázku 23.

Práce popisuje různé přístupy k výuce a modelování systémů a následně se soustředí na návrh a realizaci tří aplikací. První se věnuje logickému řízení, kde studenti mohou sledovat principy fungování logických obvodů. Druhá aplikace je zaměřena na PSD regulátor, jehož parametry lze interaktivně nastavovat a sledovat vliv na regulaci. Třetí aplikace simuluje chování dynamických systémů, což umožňuje lepší pochopení reakcí na změny vstupních podmínek [17].



Obrázek 23 Ukázka z výukové aplikace

(zdroj: [17])

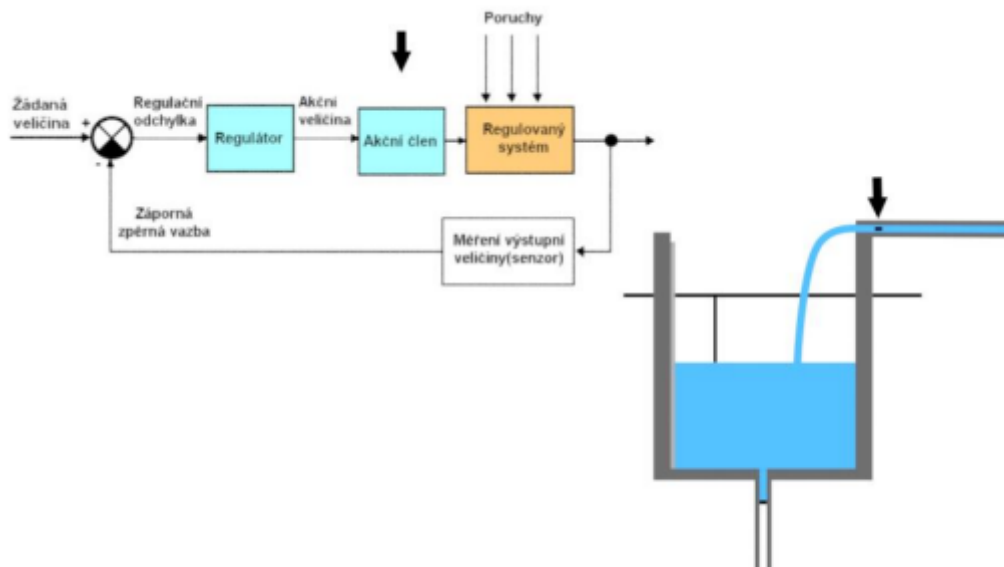
3.4 Výukové animace pro předmět Spojité řízení

Bakalářská práce Matěje Novotného se zabývá tvorbou výukových animací pro podporu výuky předmětu Spojité řízení. Cílem práce je přiblížit studentům vybrané oblasti automatického řízení pomocí animací.

Teoretická část práce popisuje základy spojitěho řízení, přenosových funkcí, regulátorů a kritérií stability. Dále se zaměřuje na dostupné nástroje pro tvorbu animací, jako jsou MATLAB, Blender nebo GIMP, a porovnává jejich vhodnost pro výukové účely.

Praktická část dokumentuje návrh, realizaci a export jednotlivých animací, které slouží jako doplněk k teoretickému výkladu. Každá animace je koncipována s ohledem na výukový přínos a tematicky se věnuje například PID regulaci, Nyquistovu kritériu či stabilitě podle Ljapunova. Na obrázku 24 je uvedena ukázka jednoho z výukových programů.

Tato práce přináší přehledný a názorný způsob, jak studentům zprostředkovat složitější pojmy spojitěho řízení formou animací, které podporují lepší porozumění látce. Výsledné animace mohou být využity jak při výuce, tak jako samostatný studijní materiál [18].



Obrázek 24 Ukázka programu

(zdroj: [18])

3.5 Multimediální podpora výuky předmětu TAŘ II – Syntéza

Bakalářská práce Martina Vičánka se věnuje problematice syntézy v oblasti automatického řízení a možnostem multimediální podpory výuky. Cílem práce bylo vytvořit srozumitelnou prezentaci klíčových témat předmětu Teorie automatického řízení II a zpřístupnit je prostřednictvím webových stránek a elektronických prezentací.

Teoretická část práce se zaměřuje na návrh a strukturu diskrétních regulátorů. Detailně jsou zde rozebrány matematické základy návrhu číslicových regulátorů, včetně použití Z-transformace, diofantických rovnic a kritérií stability. Práce obsahuje i praktické příklady návrhu regulačních obvodů.

Praktická část je zaměřena na multimediální podporu výuky. Autor navrhl a vytvořil vlastní webovou stránku s výukovým obsahem a doprovodnou prezentaci v PowerPointu. Obrázek 25 zachycuje ukázkou z výukové webové stránky.

Tato práce kombinuje odborné znalosti z oblasti syntézy řízení s praktickým přesahem do oblasti e-learningu [19].



Obrázek 25 Ukázka webových stránek

(zdroj: [19])

II PRAKTICKÁ ČÁST

4 SPECIFIKACE APLIKACE

Tato kapitola se zaměřuje na podrobný popis vytvořené aplikace, její funkčnosti, herních pravidel a postup vývoje.

4.1 Popis aplikace

Aplikace byla vytvořena v prostředí MATLAB App Designer jako interaktivní vzdělávací nástroj pro podporu výuky spojitého řízení. Primárním cílem aplikace je zábavnou formou prověřit a procvičit znalosti studentů v oblasti teorie spojitých systémů a základů práce v MATLABu. Aplikace je určena pro jednoho hráče a kombinuje prvky deskové hry s kvízovými otázkami různé obtížnosti. Hráč postupně prochází sérií otázek, které odpovídají klíčovým tématům.

Herní prostředí je stylizováno do světa populární videohry Mario, kde hlavní postava vystupuje jako zachránce princezny uvězněné na vrcholu hradu. Na základě tohoto tématu nese aplikace název **Save the Princess**.

4.2 Pravidla hry

Cílem hráče je postupně projít 25 herních polí, která tvoří pomyslné schodiště vedoucí k věži hradu. Každé z těchto polí symbolizuje jednu výukovou otázku s třemi možnými odpověďmi (A, B, C), přičemž vždy pouze jedna je správná. Otázky jsou rozděleny do tří úrovní obtížnosti – Easy, Medium a Hard.

Hráč odpovídá na otázky podle zvolené obtížnosti, přičemž se jeho herní postava posouvá po herním poli směrem vpřed v případě správné odpovědi, nebo zpět v případě odpovědi nesprávné – a to v závislosti na zvolené úrovni obtížnosti.

Posun hráče dle obtížnosti:

- **Easy** – správná odpověď: **+1 pole**, chybná odpověď: **žádný posun**
- **Medium** – správná odpověď: **+2 pole**, chybná odpověď: **-1 pole**
- **Hard** – správná odpověď: **+3 pole**, chybná odpověď: **-2 pole**

Hra je úspěšně dokončena v okamžiku, kdy hráč zodpoví otázku na posledním (25.) políčku, čímž symbolicky zachrání princeznu. Následně je automaticky vyhodnocen celkový čas a počet chybných odpovědí. Hráč je poté vyzván k zadání svého jména. Výsledek je uložen do výsledkové tabulky, která se automaticky zobrazí.

5 VÝVOJ APLIKACE

Tato kapitola se zaměřuje na popis aplikace z pohledu programátora, a to zejména na její vnitřní strukturu, použitá data a logiku jednotlivých funkcí.

5.1 Proces tvorby aplikace

Vývoj aplikace Save the Princess probíhal v prostředí MATLAB R2024b s využitím nástroje App Designer, který umožňuje tvorbu grafického uživatelského rozhraní v kombinaci s přímým napojením na kód v jednom prostředí. Hlavní výhodou tohoto přístupu je přehlednost a snadné ladění aplikace během vývoje.

Před zahájením samotného programování jsem se seznámil s nástrojem App Designer prostřednictvím veřejně dostupných aplikací a ukázkových projektů. Nastudoval jsem si základní principy jeho používání a porozuměl tomu, jak lze pomocí tohoto nástroje propojit uživatelské rozhraní s funkcionalitou aplikace. Na základě získaných poznatků jsem vytvořil první návrh vlastní aplikace, který jsem v průběhu vývoje dále rozšiřoval a upravoval podle potřeb herní logiky.

5.2 Struktura aplikace

Samotná hra je rozdělena do tří hlavních .mlapp souborů (tříd):

Menu.mlapp – úvodní obrazovka s tlačítky pro spuštění hry, zobrazení pravidel, výsledků a ukončení aplikace,

Pravidla.mlapp – samostatné okno zobrazující herní pravidla,

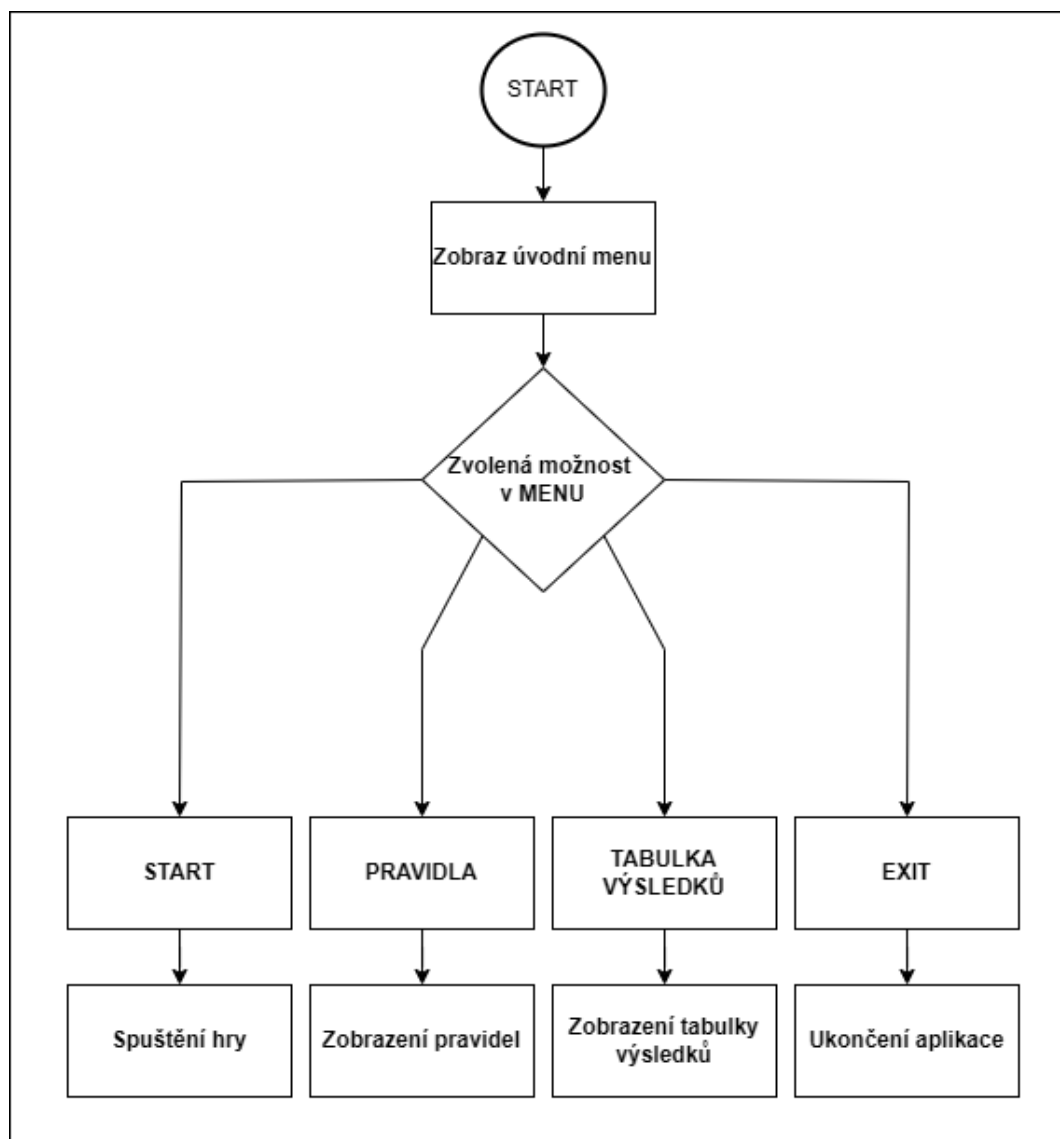
Hra.mlapp – hlavní herní logika, zobrazení otázek, řízení pohybu hráče a vyhodnocení výsledků.

5.2.1 Menu.mlapp

Třída **Menu.mlapp** tvoří úvodní obrazovku aplikace a slouží jako výchozí bod pro spuštění hry. Po spuštění aplikace se zobrazí okno obsahující čtyři tlačítka. Tato tlačítka slouží k přechodu do jednotlivých částí aplikace nebo k jejímu ukončení:

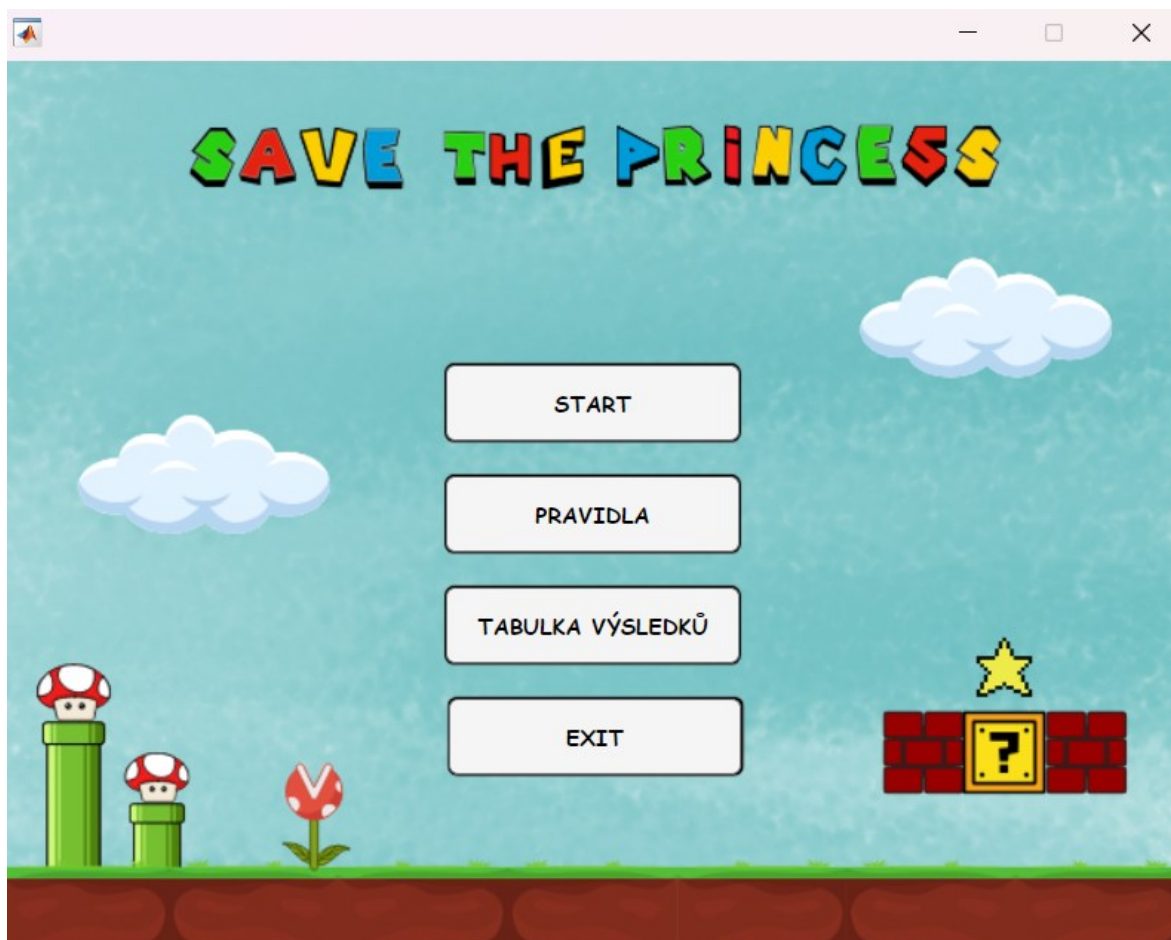
- **START** – slouží ke spuštění samotné hry. Po kliknutí se vytvoří instance třídy **Hra**, která se zobrazí na obrazovce. Současně se okno **Menu** skryje a zůstává pouze na pozadí. Do nově vytvořeného herního okna se zároveň předá odkaz na **Menu**, aby bylo možné se k němu vrátit po stisknutí tlačítka **ZPĚT** ve hře.
- **PRAVIDLA** – otevře nové okno, kde jsou uživateli zobrazena pravidla a vysvětlivky k mechanice hry.
- **TABULKA VÝSLEDKŮ** – po kliknutí na tlačítko se načte obsah souboru **leaderboard.mat**, který obsahuje uložené výsledky hráčů (jméno, dosažený čas a počet chyb). Následně se otevře nové okno s přehlednou tabulkou, v níž jsou výsledky seřazeny vzestupně podle času a doplněny o číselné pořadí.
- **EXIT** – po stisknutí tlačítka je zobrazeno potvrzovací dialogové okno s dotazem, zda si uživatel skutečně přeje aplikaci ukončit. V případě potvrzení volby se hlavní okno uzavře a aplikace se ukončí.

Vývojový diagram hlavního menu aplikace je znázorněn na obrázku 26 a jeho grafická podoba je zachycena na obrázku 27.



Obrázek 26 Vývojový diagram hlavního menu aplikace

(zdroj: vlastní)

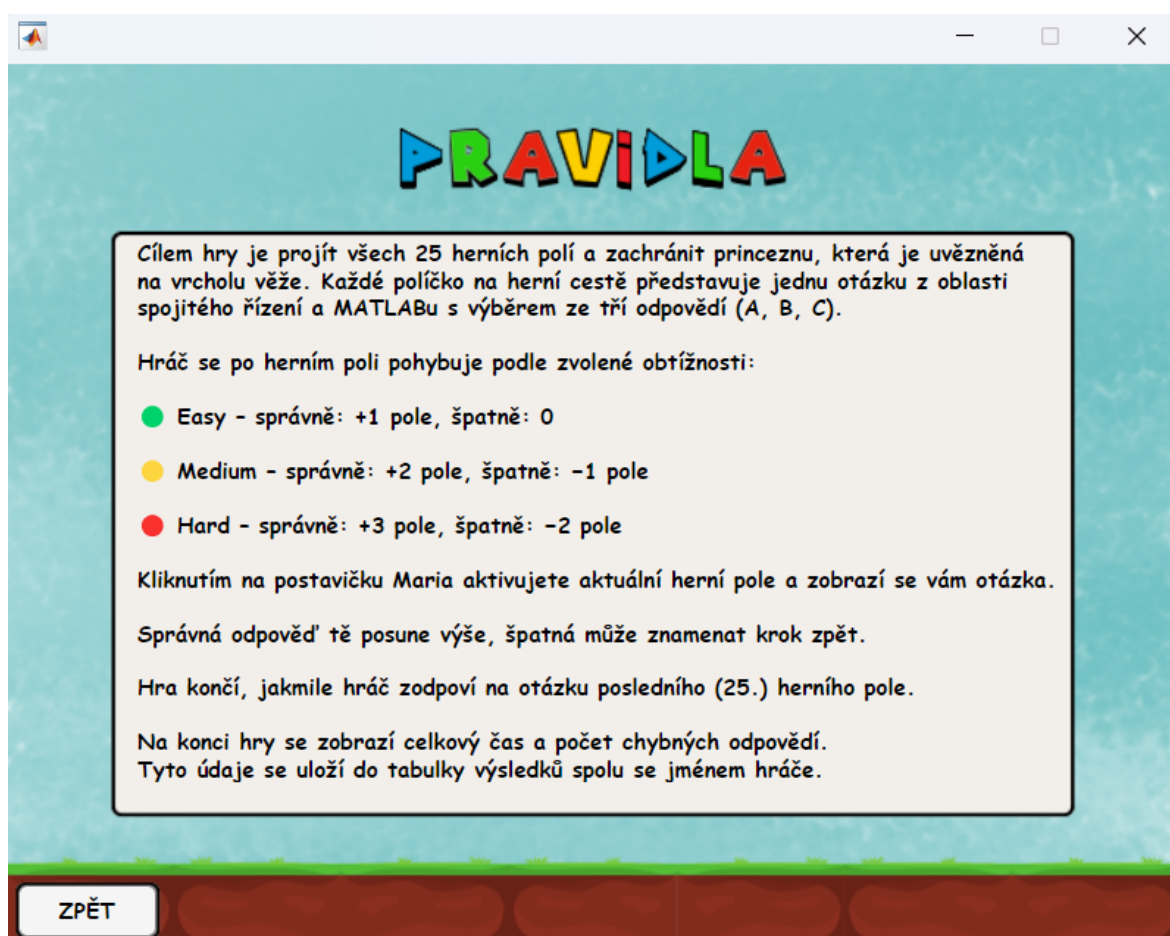


Obrázek 27 Úvodní obrazovka aplikace Save the Princess

(zdroj: vlastní)

5.2.2 Pravidla.mlapp

Třída **Pravidla** slouží k přehlednému zobrazení herních pravidel a ovládání. Její hlavní funkcí je informovat uživatele o cíli hry, způsobu pohybu po herním poli, principech obtížnosti a vyhodnocení výsledků. Jedná se o samostatné okno, které se otevře po kliknutí na tlačítko **PRAVIDLA** v hlavním menu aplikace, jak ukazuje obrázek 28.



Obrázek 28 Okno s pravidly hry Save the Princess

(zdroj: vlastní)

5.2.3 Hra.mlapp

Třída **Hra.mlapp** tvoří jádro celé aplikace a obsahuje kompletní herní logiku i většinu vizuálních a interaktivních prvků. Po spuštění se otevře nové okno s herním rozhraním, ve kterém probíhá celý průběh hry od volby obtížnosti přes postupné zodpovídání otázek až po vyhodnocení výsledků a zobrazení tabulky s dosaženými výsledky po úspěšném dokončení hry. Tato třída je nejrozsáhlejší a její kód je rozdělen do několika logických celků, které zajišťují funkčnost jednotlivých částí hry.

Klíčové komponenty:

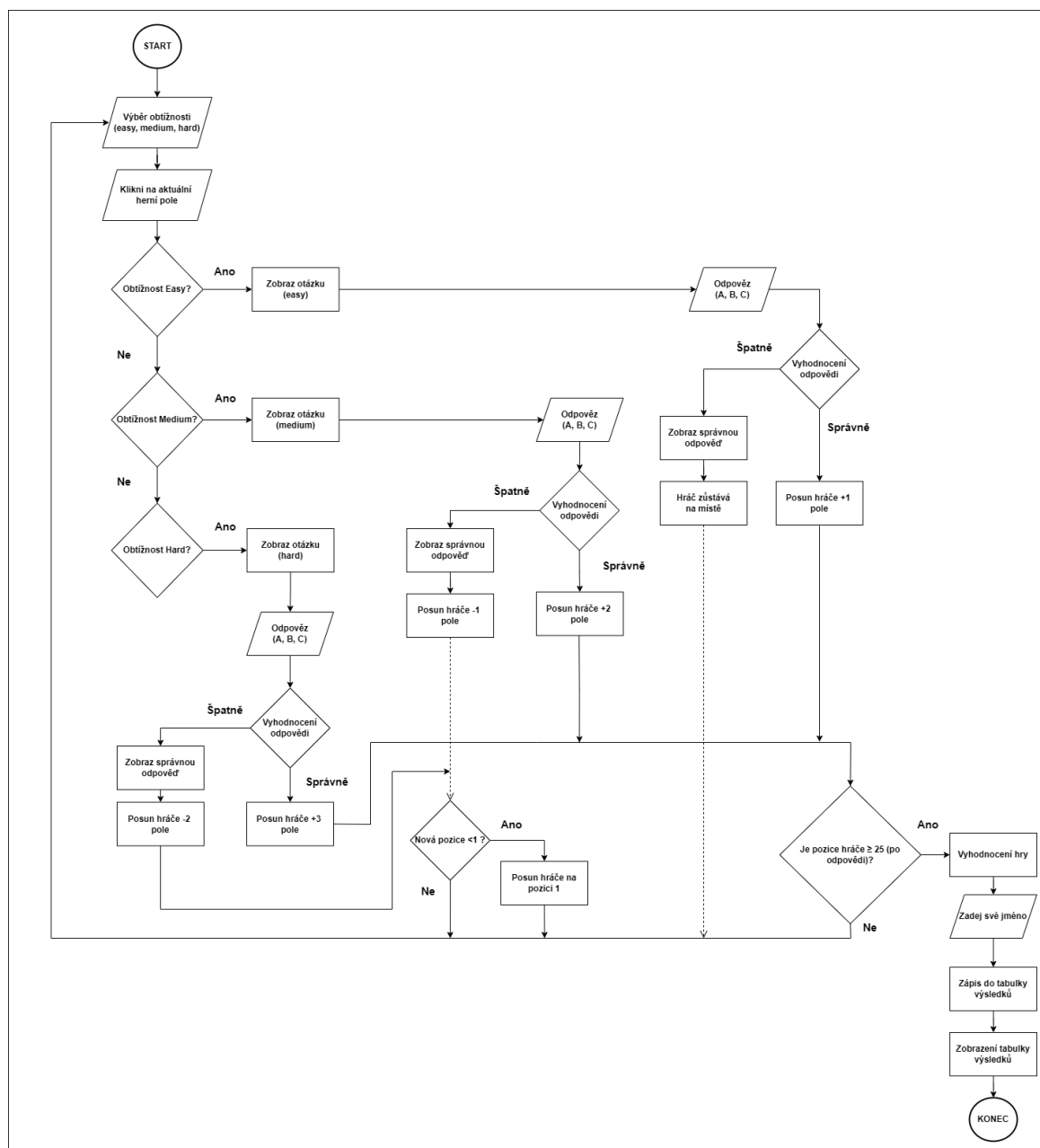
Herní pole je tvořeno 25 tlačítky, které symbolizují jednotlivá políčka schodiště. Aktivní je vždy pouze to tlačítko, na kterém se právě nachází hráč.

Postavička Maria je reprezentována jako obrázek a slouží k vizuálnímu označení aktuální pozice hráče. Kliknutím na postavičku se aktivuje příslušná otázka.

Panel s otázkou obsahuje label s textem otázky, tři odpovědi (A, B, C), tlačítko pro potvrzení odpovědi a label pro zobrazení správné odpovědi v případě chyby. Výběr obtížnosti je realizován pomocí přepínače (Easy, Medium, Hard). Výběr ovlivňuje pravidla pohybu hráče po hracím poli. Ovládací tlačítka **ZPĚT** pro návrat do menu, **RESET** pro restart hry, **EXIT** pro ukončení aplikace.

Na obrázku 29 je znázorněn vývojový diagram herní logiky, který přehledně zachycuje průběh hry od výběru obtížnosti přes odpovídání na otázky až po vyhodnocení a zobrazení výsledků.

Na obrázku 30 je zobrazeno herní okno aplikace, ve kterém se hráč pohybuje po jednotlivých políčkách herního pole reprezentovaných tlačítky.



Obrázek 29 Vývojový diagram herní logiky

(zdroj: vlastní)



Obrázek 30 Herní okno aplikace

(zdroj: vlastní)

5.3 Klíčové funkce a herní logika

Tato kapitola se věnuje procesům, které v aplikaci probíhají během celé hry z pohledu jejího fungování.

5.3.1 startupFcn

Funkce `startupFcn` slouží jako výchozí bod aplikace, který se automaticky vykoná po spuštění herního okna (**Hra.mlapp**). Jejím hlavním úkolem je připravit celé herní rozhraní tak, aby bylo plně funkční a vizuálně připravené na zahájení hry. Tato funkce sama o sobě neprovádí všechny kroky přímo, ale volá několik funkcí, které zajišťují jednotlivé části inicializace. Konkrétně se jedná o:

- **centerFigure** – vycentrování okna aplikace na obrazovce
- **SetupImage** – nastavení grafického pozadí a postavičky Maria

- **LoadQuestions** – načtení otázek z externího Excel souboru
- **SetupButtons** – nastavení herních tlačítek a jejich vizuálního stavu
- **UpdateMarioPosition** – umístění postavičky Maria na výchozí políčko.

Dále jsou v rámci této funkce nastaveny výchozí hodnoty proměnných, jako je počáteční pozice hráče (**PlayerPosition**), celkový počet polí (**TotalFields**) a viditelnost ovládacích prvků. Funkce **startupFcn** je tak klíčovým prvkem pro správné spuštění a bezchybnou inicializaci celé hry.

```
function startupFcn(app, Menu)
    app.UIFigure.Visible = 'off';
    app.MenuHandle=Menu; % Odkaz na Menu
    centerFigure(app.UIFigure); % Vystředění hlavního okna
    app.UIFigure.Resize = 'off';
    app.UIFigure.Name = '';

    SetupImage(app); %Nastavení obrázku pozadí a Maria
    LoadQuestions(app); %Načtení otázek z excelu

    app.PlayerPosition = 1; % Nastavení výchozí pozice
    app.TotalFields = 25; %Nastavení počtu herních polí
    app.Panel.Visible = 'off';
    app.correctLabel.Visible="off";
    SetupButtons(app); Funkce nastavení tlačítek
    UpdateMarioPosition(app); % Funkce aktualizace pozice Maria
    app.UIFigure.Visible = 'on';
end
```

Zdrojový kód 1 Funkce startupFcn

5.3.2 centerFigure.m

Funkce **centerFigure** slouží k automatickému vycentrování okna aplikace na střed obrazovky. Je volána při spuštění všech oken aplikace, aby bylo každé z oken zobrazeno uprostřed. Tato funkce je vytvořena jako samostatný .m soubor, aby se nemusel stejný kód opakovat ve všech třídách. Funkce obsahuje vstupní argument **figHandle**, což je odkaz na konkrétní grafické okno (**UIFigure**). Na základě systémových údajů o velikosti obrazovky a aktuálních rozměrů okna vypočítá nové souřadnice levého horního rohu tak, aby bylo okno přesně vycentrováno.

```
function centerFigure(figHandle)
    drawnow;
    screenSize = get(0, 'ScreenSize');
    figWidth = figHandle.Position(3);
    figHeight = figHandle.Position(4);
    figHandle.Position(1:2) = [ ...
        (screenSize(3) - figWidth) / 2, ...
        (screenSize(4) - figHeight) / 2 ...
    ];
end
```

Zdrojový kód 2 Funkce centerFigure

5.3.3 SetupImage

Funkce **SetupImage** slouží k nastavení pozadí herního okna a přidání postavičky hráče – Maria. Na začátku funkce se vytvoří grafický objekt typu **uiimage**, kterému je přiřazen obrázek pozadí (**pole.png**). Tento obrázek je následně umístěn na pozadí okna a pomocí funkce (**uistack**) se přesune do spodní vrstvy, aby nepřekrýval žádné herní komponenty. Následně se vytvoří druhý grafický objekt – postavička Maria (**mario.png**). Po spuštění hry se přesune na první políčko. Postavička je interaktivní – při kliknutí na ni se aktivuje aktuální otázka. Tuto interakci zajišťuje přiřazení funkce **MarioClicked** pomocí **ImageClickedFcn**.

```
function SetupImage(app)
    % Nastavení pozadí
    background = uiimage(app.UIFigure); % obrázek
    background.ImageSource = 'pole.png'; % cesta k obrázku
    background.Position = [1 1 926 748]; % rozměr okna
    background.ScaleMethod = 'none'; % roztahání
    uistack(background, 'bottom'); % obrázek bottom layer
    % Vytvoření obrázku Maria
    app.MarioImage = uiimage(app.UIFigure);
    app.MarioImage.ImageSource = 'mario.png'; % název souboru
    app.MarioImage.Position = [-100 -100 71 55];
    app.MarioImage.Visible = 'on';
    % spuštění funkce při kliknutí na maria
    app.MarioImage.ImageClickedFcn = @(src, event) MarioClicked(app);
    uistack(app.MarioImage, 'top'); % Mario nad tlačítky
    uistack(app.Panel, 'top'); % Panel nad Mariem
end
```

Zdrojový kód 3 Funkce SetupImage

5.3.4 LoadQuestions

Funkce **LoadQuestions** slouží k načtení otázek z Excel souboru **questions.xlsx**. Tento Excel soubor obsahuje sadu otázek rozdělených podle obtížnosti, které se následně používají během hry. Po spuštění funkce se nejprve zjistí úplná cesta k souboru, poté se zjistí formát dat v tabulce, a následně se nastaví vybrané sloupce na datový typ *string*, aby bylo možné s daty pracovat jako s textem. Po načtení dat do proměnné **data** se otázky rozdělí do tří skupin podle obtížnosti – easy, medium a hard. Každá skupina se uloží jako struktura do pole (**app.Questions**). Každá z obtížností má dvě varianty:

- **.All** – obsahuje všechny načtené otázky této obtížnosti
- **.Remaining** – obsahuje otázky, které ještě nebyly v dané hře použity

Toto rozdělení má praktický význam při náhodném výběru otázky během hry. Výběr a postupné odstraňování otázek ze seznamu **Remaining** pak probíhá ve funkci (**FieldClicked**). Pokud v průběhu hry dojde k vyčerpání všech otázek dané obtížnosti, seznam **Remaining** se automaticky obnoví ze seznamu **All**, aby bylo možné hru plynule dokončit.

```
function LoadQuestions(app)
% Načtení otázek z Excelu
%úplná cesta k souboru (v aktuální složce)
    QuestionsFile = fullfile(files(), "questions.xlsx");
    opts = detectImportOptions(QuestionsFile); % zjištění struktury
        excelu
    opts = setvartype(opts, {'text','A','B','C','correct','obtiznost'},
        'string'); %Převedení na string
    data = readtable(QuestionsFile, opts); % Načtení excelu do proměnné
% Rozdělení otázek podle obtížnosti
    app.Questions.easy.All = table2struct(data(strcmp(data.obtiznost,
        "easy"), :));
    app.Questions.medium.All = table2struct(data(strcmp(data.obtiznost,
        "medium"), :));
    app.Questions.hard.All = table2struct(data(strcmp(data.obtiznost,
        "hard"), :));
% Nahrání všech otázek pro následné odstranění již použitých otázek
    app.Questions.easy.Remaining = app.Questions.easy.All;
    app.Questions.medium.Remaining = app.Questions.medium.All;
    app.Questions.hard.Remaining = app.Questions.hard.All;
end
```

5.3.5 SetupButtons

Funkce **SetupButtons** slouží k nastavení všech herních tlačítek, která tvoří jednotlivé herní pole (celkem 25). V rámci této funkce se provádí nastavení vzhledu a textu tlačítka. Aktivní zůstává pouze první tlačítko (**Button_1**), ostatní jsou na začátku hry skryté a neaktivní. Každému tlačítku se přiřadí funkce (**FieldClicked**), která zobrazí otázku při kliknutí na dané políčko. Toto přiřazení probíhá pomocí cyklu, čímž se vyhneme ručnímu nastavení pro každé tlačítko zvlášť.

```
function SetupButtons(app)
    % Uvodní nastavení tlačítek
    for i = 1:app.TotalFields
        FieldName = "Button_" + i;
        btn = app.(FieldName);

        % Nastavení stylu tlačítek
        btn.Text = '';
        btn.BackgroundColor = app.UIFigure.Color;

        % Zobrazit jen první tlačítko
        btn.Visible = strcmp(FieldName, "Button_1");
        btn.Enable = strcmp(FieldName, "Button_1");

        % Přiřazení callback funkce tlačítkům
        btn.ButtonPushedFcn = @(src, event) FieldClicked(app, i);
    end
end
```

Zdrojový kód 5 Funkce SetupButtons

5.3.6 UpdateMarioPosition

Funkce **UpdateMarioPosition** slouží k zobrazení Maria nad aktuálním herní pole, na kterém se hráč nachází. Funkce funguje tak, že na základě aktuální pozice hráče (**PlayerPosition**) sestaví název odpovídajícího tlačítka. Následně z tohoto tlačítka načte pozici a tu přiřadí obrázku Maria. Díky tomu se Mario vždy přesune na správné herní pole.

```
function UpdateMarioPosition(app)
    % Posun maria na aktuální pozici
    % sestavení názvu aktuálního tlačítka
    CurrentField = "Button_" + app.PlayerPosition;

    btn = app.(CurrentField); % přiřzení tlačítka
    Position = btn.Position; %pozice tlačítka

    % Nastavení pozice Maria na pozici tlačítka
    app.MarioImage.Position = Position;
end
```

Zdrojový kód 6 Funkce UpdateMarioPosition

5.3.7 MarioClicked

Funkce **MarioClicked** slouží k aktivaci aktuálního herního pole poté, co uživatel klikne na postavičku Maria. Mario je ve hře použit jako vizuální reprezentace pozice hráče, a protože překrývá aktivní tlačítko na herním poli, není možné na tlačítko kliknout přímo. Místo toho se kliknutí na Maria přesměruje na příslušné tlačítko pomocí této funkce.

```
function MarioClicked(app)
    %Kliknutí na tlačítko pod obrázkem Maria
    % sestavení názvu aktuálního tlačítka
    CurrentField = "Button_" + app.PlayerPosition;
    btn = app.(CurrentField); % přiřazení tlačítka
    btn.ButtonPushedFcn(btn, []); % Aktivace kliknutí na tlačítko
end
```

Zdrojový kód 7 Funkce MarioClicked

5.3.8 FieldClicked

Funkce **FieldClicked** je volána pokaždé, když hráč klikne na herní pole. Jejím úkolem je připravit a zobrazit otázku odpovídající aktuálně zvolené obtížnosti. Funkce zároveň provádí výběr otázky, zamíchání odpovědí, nastavení vizuálních prvků a uložení všech dat potřebných pro pozdější vyhodnocení odpovědi. Protože je tato funkce poměrně rozsáhlá, v této kapitole jsou popsány jen její nejdůležitější části. Ty jsou přehledně rozděleny do několika logických bloků, podle toho, jak jednotlivé kroky navazují v reálném průběhu hry:

1. **Spuštění časovače** – funkce spustí časovač, pokud se jedná o první pole a zda ještě nebyl doposud spuštěn.

```
%spuštění timeru při kliknutí na první tlačítko
if index == 1 && all(isnat(app.StartTime))
    app.StartTime = datetime("now");
end
```

Zdrojový kód 8 Funkce FieldClicked – spuštění časovače

2. **Obtížnosti a obnova otázek** – načte se zvolená obtížnost ze skupiny radio buttonů. Pokud dojdou otázky této obtížnosti, seznam **Remaining** se obnoví ze seznamu **All**

```
% Načtení obtížnosti z radio buttonu
Difficulty = app.ObtnostButtonGroup.SelectedObject.Text;

% Pokud už jsou vyplýtvány všechny otázky, obnoví se
if isempty(app.Questions.(Difficulty).Remaining)
    app.Questions.(Difficulty).Remaining =
    app.Questions.(Difficulty).All;
end
```

Zdrojový kód 9 Funkce FieldClicked – obtížnost a obnova otázek

3. **Náhodný výběr otázky** – otázka je náhodně vybrána a následně odstraněna ze seznamu (**Remaining**), aby se v aktuální hře již neopakovala

```
questions = app.Questions.(Difficulty).Remaining;
idx = randi(numel(questions));
q = questions(idx);
% Odebere se vybraná otázku ze seznamu Remaining
questions(idx) = [];
app.Questions.(Difficulty).Remaining = questions;
```

Zdrojový kód 10 Funkce FieldClicked – náhodný výběr otázky

4. **Zobrazení otázky a zamíchání odpovědí** – otázka se zobrazí v labelu. Následně se odpovědi zamíchají, aby nebyly vždy ve stejném pořadí

```
% Zobrazení otázky v labelu
app.otazkaLabel.Text = q.text;

% Shuffle odpovědí
Options = {q.A, q.B, q.C}; % původní možnosti
Labels = {'A', 'B', 'C'}; % označení
random = randperm(3); % zamíchání 1-3

% náhodně přiřazení odpovědi k radiobuttonu A), B), C)
ShuffledOptions = Options(random);
```

Zdrojový kód 11 Funkce FieldClicked – zamíchání odpovědí

Texty jednotlivých odpovědí se podle nového pořadí přiřadí k **radiobuttonům**.

```
% Aktualizace textu tlačítek
app.aButton.Text = "A) " + ShuffledOptions{1};
app.bButton.Text = "B) " + ShuffledOptions{2};
app.cButton.Text = "C) " + ShuffledOptions{3};
```

Zdrojový kód 12 Funkce FieldClicked – přiřazení otázky k radiobuttonům

5. **Uložení správné odpovědi** – funkce si zapamatuje, která odpověď je po zamíchání tou správnou, aby ji bylo možné později správně vyhodnotit

Funkce si nejprve načte informaci o tom, která odpověď byla ve zdrojových datech označena jako správná.

```
% Přiřazení správných odpovědí po zamíchání
original = q.correct; %původní

% původní pozice správné odpovědi
CorrectIndex = find(strcmp(Labels, original));
```

Zdrojový kód 13 Funkce FieldClicked – původní správná odpověď

Následně zjistí, na jaké nové pozici se tato odpověď nachází po náhodném zamíchání.

```
% Zjištění pozice přesunuté správné odpovědi
NewCorrectIndex = random == CorrectIndex;

% nové písmeno správné odpovědi ('A', 'B', 'C')
Correct = Labels{NewCorrectIndex};

% výběr původní správné odpovědi
correctAnswerText = ShuffledOptions{random == CorrectIndex};
```

Zdrojový kód 14 Funkce FieldClicked – zjištění nové pozice

6. **Uložení dat do panelu** – všechna důležitá data o aktuální otázce se uloží do vlastnosti **UserData** u panelu, aby je bylo možné použít při vyhodnocování odpovědi:

```
%uložení dat do panelu
app.Panel.UserData = struct( ...
    "ClickedField", index, ...
    "Correct", Correct, ...
    "CorrectText", correctAnswerText, "Question", q);
```

Zdrojový kód 15 Funkce FieldClicked – uložení dat do panelu

5.3.9 OdpovdtButtonPushed

Funkce **OdpovdtButtonPushed** se spustí po kliknutí na tlačítko **odpovědět**. Zjistí, kterou odpověď hráč vybral, a podle zvolené obtížnosti nastaví, o kolik polí se má hráč posunout vpřed nebo vzad. Následně porovná odpověď hráče se správnou odpovědí a podle výsledku zavolá buď funkci **CorrectAnswer**, nebo **WrongAnswer**.

```
function OdpovdtButtonPushed(app, event)
    data = app.Panel.UserData; %načtení dat uložených v panelu
    % Zjištění vybrané odpovědi
    selected = app.OdpovButtonGroup.SelectedObject;
    if isempty(selected)
        uialert(app.UIFigure, "Vyber odpověď!", "Chyba");
        return;
    end

    %Vyčtení prvního znaku odpovědi
    UserAnswer = selected.Text(1);
```

```
%zjištění obtížnosti
difficulty = app.ObtностButtonGroup.SelectedObject.Text;

% Posun podle obtížnosti
switch difficulty
    case "easy"
        CorrectStep = 1;
        WrongStep = 0;
    case "medium"
        CorrectStep = 2;
        WrongStep = -1;
    case "hard"
        CorrectStep = 3;
        WrongStep = -2;
    otherwise
        CorrectStep = 1;
        WrongStep = 0;
end

% Porovnání hráčovy odpovědi se správnou odpovědí a vyhodnocení
if strcmpi(UserAnswer, data.Correct)
    CorrectAnswer(app, data, CorrectStep);
else
    WrongAnswer(app, data, WrongStep);
end

% Skrytí panelu s otázkou
app.Panel.Visible = 'off';
app.UIFigure.Visible = 'on';
end
```

Zdrojový kód 16 Funkce OdpovdtButtonPushed

5.3.10 CorrectAnswer

Funkce **CorrectAnswer** se volá v okamžiku, kdy hráč odpoví správně. Jedná se o jednu z nejdůležitějších funkcí celé aplikace, protože kromě posunu hráče po herním poli také kontroluje, zda hráč hru dokončil, a v případě úspěchu vyhodnotí výsledky a zobrazí tabulku. Protože je funkce poměrně rozsáhlá, jsou níže popsány pouze klíčové kroky kódu:

1. **Zobrazení hlášky a posun hráče vpřed** – Hráči se zobrazí potvrzení správné odpovědi formou dialogového okna. Následně se hráč posune vpřed podle nastavené obtížnosti.

```
msg = msgbox("Správně!");  
uiwait(msg);  
  
%Posun hráče vpřed podle obtížnosti  
app.PlayerPosition = min(app.TotalFields, data.ClickedField +  
CorrectStep);  
UpdateMarioPosition(app); % Aktualizace pozice Maria
```

Zdrojový kód 17 Funkce CorrectAnswer – zobrazení hlášky a posun hráče vpřed

2. **Zabarvení políček** – Všechna pole, které již hráč prošel, se zbarví zeleně a deaktivují se, aby na ně nebylo možné znovu kliknout. Nové pole, na kterém se aktuálně nachází, se zvýrazní a aktivuje.

```
%Zbarvení všech tlačítek, mimo aktuální na zeleno  
for j = 1:app.PlayerPosition - 1  
    FieldName = "Button_" + j;  
    app.(FieldName).Visible = 'on';  
    app.(FieldName).BackgroundColor = [0.6, 1.0, 0.6]; % zelená  
    app.(FieldName).Enable = 'off'; %zakázání kliknutí  
end  
  
%Zobrazení aktuálního tlačítka na které je hráč  
if app.PlayerPosition < app.TotalFields  
    CurrentField = "Button_" + app.PlayerPosition;  
    app.(CurrentField).Visible = 'on';  
    app.(CurrentField).BackgroundColor = [0.94, 0.94, 0.94];  
    app.(CurrentField).Enable = 'on'; % Povolení kliknutí  
end
```

Zdrojový kód 18 Funkce CorrectAnswer – zabarvení políček

3. **Detekce konce hry** – Pokud hráč odpovědí dosáhne nebo překročí poslední (25.) políčko, hra je úspěšně dokončená. Ukončí se časomíra, vypočítá se celkový čas a hráč je vyzván k zadání jména, které se spolu s výsledkem uloží do tabulky.

```
% Pokud je zodpovězená otázka na posledním políčku (25)
if app.PlayerPosition >= app.TotalFields
app.EndTime = datetime("now"); % Ukončení timeru
time = seconds(app.EndTime - app.StartTime); %Výpočet celkového času

%Výherní zpráva s výsledky
msg = sprintf("Gratulujeme! Dokončil jsi hru!\n Čas: %.2f s\n Chyby:
%d", time, app.WrongAnswers);
msg = msgbox(msg, "Konec hry");
uiwait(msg);

% Zadání jména
Name = inputdlg("Zadej své jméno:", "Dokončeno", [1 50]);
if isempty(Name)
    Name = "Neznámý hráč";
else
    Name = Name{1}; %Přiřazení záznamu z pole
if strlength(Name) > 15
    Name = extractBefore(Name, 16); % pokud je delší než 15 zkrát
end
end
```

Zdrojový kód 19 Funkce CorrectAnswer – detekce konce hry

4. **Záznam výsledku a zobrazení tabulky** – Výsledek se uloží do souboru **leaderboard.mat**, který slouží jako tabulka výsledků. Pokud soubor neexistuje, vytvoří se nový. Výsledky všech hráčů se seřadí podle času a zobrazí se v novém okně ve formě jednoduché tabulky.

```
%Nový záznam hráče
NewRecord = struct("Jmeno", Name, "Cas", time, "Chyby",
    app.WrongAnswers);

%Načtení tabulky, pokud neexistuje, vytvoření nové
LeaderboardFile = fullfile(files(), "leaderboard.mat");
if isfile(LeaderboardFile)
    data = load(LeaderboardFile);
    if isfield(data, "Leaderboard")
        Leaderboard = data.Leaderboard;
    else
        Leaderboard = [];
    end
else
    Leaderboard = [];
end

%Zapsání záznamu do tabulky a uložení
Leaderboard = [Leaderboard; NewRecord];
save(LeaderboardFile, "Leaderboard");

[~, idx] = sort([Leaderboard.Cas]); %seřazení záznamů podle času
Leaderboard = Leaderboard(idx); %přeuspořádání podle indexu

%vytvoření hlavičky tabulky
head = sprintf(" # %-15s %-10s %-5s", "Jméno", "Čas [s]",
    "Chyby");
divide = repmat('-', 1, 49); %oddělovací čára
rows = strings(1, numel(Leaderboard) + 2);
rows(1) = head;
rows(2) = divide;
%odřádkování
for i = 1:numel(Leaderboard)
    p = Leaderboard(i);
    rows(i+2) = sprintf(" %-2d %-15s %8.2f %6d", i, p.Jmeno, p.Cas,
        p.Chyby);
end
```

```

leaderboardMsg = strjoin(rows, newline); %spojení dohromady

%Vytvoření nového okna, načtení výsledků a zavření
Table = uifigure('Name', 'Leaderboard', 'Position', [100 100 400 300]);
movegui(Table, 'center');
uitextarea(Table, ...
'Value', splitlines(leaderboardMsg), ...
'FontName', 'Courier New', ...
'Editable', 'off', ...
'Position', [10 50 380 230]);
uibutton(Table, ...
'Text', 'Zavřít', ...
'Position', [160 10 80 30], ...
'ButtonPushedFcn', @(btn, event) close(Table));
end

```

Zdrojový kód 20 Funkce CorrectAnswer – záznam výsledku a zobrazení tabulky

5.3.11 WrongAnswer

Funkce **WrongAnswer** se volá ve chvíli, kdy hráč odpoví špatně. Jejím hlavním cílem je vyhodnotit chybnou odpověď, zvýšit čítač chyb, zobrazit správné řešení a posunout hráče zpět podle zvolené obtížnosti. Funkce zároveň upravuje herní políčka, aby hráč viděl svůj aktuální postup i chybnou odpověď.

```

function WrongAnswer(app, data, wrongStep)
% ošetření špatné odpovědi
    app.WrongAnswers = app.WrongAnswers + 1; % čítač špatných odpovědí

% zobrazení správné odpovědi
    Correct = data.Correct + ") " + data.CorrectText;
    app.correctLabel.Text = "Správná odpověď: " + Correct;
    app.correctLabel.BackgroundColor = [0.6, 1.0, 0.6];
    app.correctLabel.Visible = 'on';

    h = msgbox("Špatně. Zkus to znovu.");
    uiwait(h);
    app.correctLabel.Visible = 'off';

% Posun hráče zpět, ale ne mimo startovní pozici
    app.PlayerPosition = max(1, app.PlayerPosition + wrongStep);
    UpdateMarioPosition(app); % aktualizace pozice Maria

```

```

%Nastavení sprvně zodpovězených polí
for j = 1:app.PlayerPosition
    FieldName = "Button_" + j;

    app.(FieldName).Visible = 'on';
    app.(FieldName).BackgroundColor = [0.6, 1.0, 0.6]; %zelená
    app.(FieldName).Enable = 'off';
end
%Nastavení špatně zodpovězených polí
CurrentField = "Button_" + app.PlayerPosition;
app.(CurrentField).Visible = 'on';
app.(CurrentField).BackgroundColor = [1.0, 0.6, 0.6]; % červená
app.(CurrentField).Enable = 'on';
end

```

Zdrojový kód 21 Funkce WrongAnswer

5.3.12 RESETButtonPushed

Funkce **RESETButtonPushed** je callback funkce, která se spustí po kliknutí na tlačítko **RESET**. Jejím účelem je kompletně resetovat aktuální průběh hry a vrátit vše do výchozího stavu, aniž by bylo nutné aplikaci znovu spouštět.

```

function RESETButtonPushed(app, event)
% Reset pozice hráče a chyb
    app.PlayerPosition = 1;
    app.WrongAnswers = 0;
    app.StartTime = NaT;
    app.EndTime = NaT;

% Skrytí všech tlačítek
    for j = 1:app.TotalFields
        FieldName = "Button_" + j;
        if isprop(app, FieldName)
            app.(FieldName).Visible = 'off';
            app.(FieldName).Enable = 'off';
            app.(FieldName).BackgroundColor = [0.94, 0.94, 0.94]; % standardní
        end
    end
end

```

```
% Zobrazení a aktivace prvního tlačítka
if isprop(app, "Button_1")
    app.Button_1.Visible = 'on';
    app.Button_1.Enable = 'on';
    app.Button_1.BackgroundColor = [0.94, 0.94, 0.94];
end

% Skrytí panelu s otázkami
app.Panel.Visible = 'off';
UpdateMarioPosition(app);
end
```

Zdrojový kód 22 Funkce RESETButtonPushed

5.3.13 ZPTButtonPushed a EXITButtonPushed

Funkce **ZPTButtonPushed** slouží k návratu z herního okna zpět do hlavního menu.

Funkce **EXITButtonPushed** vyvolá potvrzovací dialogové okno, které se uživatele zeptá, zda si skutečně přeje aplikaci ukončit. Pokud zvolí možnost Ano, okno aplikace se zavře a hra se ukončí.

5.4 Datové soubory

Aplikace pracuje se dvěma soubory, které slouží k načítání otázek a ukládání výsledků hráčů. Díky tomu lze snadno měnit obsah otázek bez nutnosti upravovat kód a zároveň uchovávat výsledky i po opakovaném spuštění hry.

5.4.1 questions.xlsx

Soubor **questions.xlsx** obsahuje databázi 150 otázek, které jsou ve hře zobrazovány. Jedná se o běžný soubor Excel, který je rozdělen do několika sloupců:

- **text** – samotné znění otázky
- **A, B, C** – jednotlivé možnosti odpovědí
- **correct** – označení správné odpovědi (A, B, C)
- **obtiznost** – obtížnost otázky (easy, medium, hard)

Struktura otázek v Excelu můžeme vidět na obrázku 31.

1	text	A	B	C	correct	obtízno
2	Co znamená, že je signál spojitý?	Mění se plynule v čase	Je tvořen jen číslicemi 0 a 1	Mění se pouze v určitých časových okamžicích	A	easy
3	Co je to nespojitý (diskrétní) signál?	Signál, který má nekonečně malou amplitudu	Signál, který je definován pouze v určitých časech	Signál, který se mění plynule v čase	B	easy
4	Co je to řízení v kontextu automatizace?	Proces, při kterém sledujeme vývoj systému	Proces, kdy se informace pouze ukládají pro pozdější použití	Proces ovládání systému za účelem dosažení určitého cíle	C	easy
5	Jaký je hlavní rozdíl mezi manuálním a automatickým řízením?	Manuální řízení provádí člověk, zatímco automatické řízení je pomalejší než manuální	Automatické řízení je pomalejší než manuální	Manuální řízení využívá počítače, zatímco automatické řízení ne	A	easy
6	Který z následujících příkladů představuje manuální řízení?	Termostat regulující teplotu	Řízení auta řidičem	Automatický tempomat v autě	B	easy
7	Co je typickým znakem přímovazebního řízení?	Vstup se upravuje podle chyby	Systém reaguje podle zpětné vazby	Výstup systému není měřen	C	easy
8	Který z následujících popisů nejlépe vystihuje zpětnovazební řízení?	Výstup systému je měřen a podle něj se upravuje vstup	Systém se chová náhodně bez vstupu	Vstup systému je určen předem a nemění se	A	easy
9	Jak se nazývá řízení, při kterém se výstup systému předpovídá?	Předpovědní řízení	Zpětnovazební řízení	Ruční řízení	B	easy
10	Jak se nazývá rozdíl mezi požadovaným a měřeným výstupem?	Odchylka výkonu	Chyba řízení	Regulační odchylka	C	easy
11	Co je hlavní funkcí akčního členu ve zpětnovazebním řízení?	Ovlivňuje řídicí veličinu vstupující do systému	Sleduje výstup systému	Vypočítává regulační odchylku	A	easy
12	Jakou veličinu označuje písmeno "u"?	Výstupní veličinu	Akční veličinu, vstup do řízeného systému	Porucha v systému	B	easy
13	Které písmeno označuje výstupní veličinu systému?	e	u	y	C	easy
14	Jaká veličina je označena jako "e" ve zpětnovazebním řízení?	Regulační odchylka	Porucha	Akční vstup	A	easy
15	Jaká veličina je označena jako "w" ve zpětnovazebním řízení?	Výslednou chybu	Požadovanou (žádanou) veličinu	Rušivý signál	B	easy
16	Co znamená obrazový přenos v teorii systémů?	Poměr amplitud signálů	Grafické zobrazení odezvy	Poměr Laplaceových obrazů výstupu a vstupu	C	easy
54	Co je hlavní úlohou derivací složky (D) v PID regulaci?	Zajišťuje odstranění statické regulační odchylky	Ovlivňuje rychlost reakce soustavy na poruchu	Zesiluje nízkofrekvenční složky vstupního signálu	B	medium
55	Co vyjadřuje pojem dopravní zpoždění v regulovaném systému?	Čas potřebný pro ustálení výstupní veličiny	Míra tlumení oscilací při změně vstupu	Zpoždění mezi okamžikem změny na vstupu a okamžikem změny na výstupu	C	medium
56	Systém je plně fideletní, pokud:	Lze stav systému dostat do libovolného stavu	Je možné ovlivnit každý stav výstupní veličiny	Je možné systém fideletně reprodukovat pouze ve stabilní oblasti	A	medium
57	Co platí o stabilitě spojitých systémů?	Systém je stabilní, pokud všechny póly leží v levé poloze	Všechny póly systému musí ležet v levé poloze	Stabilita závisí pouze na počátečních podmínkách	B	medium
58	Jaké je hlavní omezení algebraických kritérií stability?	Neposkytují přesnou hodnotu přenosové funkce	Nelze je použít na spojitých systémech	Nejsou vhodná pro systémy s dopravním zpožděním	C	medium
59	Co znamená, že systém je pozorovatelný?	Že lze z výstupní veličiny určit všechny vnitřní stavy	Že lze určit všechny vstupy systému	Že systém má reálné koeficienty v přenosové funkci	A	medium
60	Jaká je funkce proporcionálního členu v regulátoru?	Předpovídá budoucí hodnoty výstupu	Zajišťuje korekci na základě velikosti regulační odchylky	Odstaňuje trvalou regulační odchylku	B	medium
61	Co zobrazuje Nyquistova charakteristika u spojitých systémů?	Průběh amplitudy výstupu v závislosti na frekvenci vstupu	Zpoždění mezi vstupem a výstupem v časové doméně	Změnu velikosti a fáze výstupu v závislosti na frekvenci vstupu	C	medium
62	K čemu slouží Nyquistovo kritérium?	K posouzení stability uzavřené smyčky na základě polohy pólů	K určení přesnosti regulace	K posouzení stability otevřené smyčky	A	medium
63	Co vyjadřuje frekvenční charakteristika systému?	Odezvu systému na náhodné vstupy	Chování systému při různých frekvencích vstupu	Přenosovou funkci v čase	B	medium
64	Co zobrazuje fázová část frekvenční charakteristiky?	Rozdíl v amplitudě mezi vstupem a výstupem	Zobrazení reálné části frekvenční odezvy	Úhlový posun výstupu vůči vstupu při sinusovém vstupu	C	medium
65	Jak se posuzuje stabilita systému z Bodeho diagramu?	Podle fázového posunu v bodě, kde má amplitudová charakteristika nulovou hodnotu	Podle velikosti zisku při nulové frekvenci	Podle počtu nul v amplitudové charakteristice	A	medium
66	Co znamená BIBO stabilita systému?	Systém má záporné vlastní čísla	Každý omezený vstup způsobí omezený výstup	Výstup systému po čase vždy klesá na nulu	B	medium

Obrázek 31 Struktura otázek v Excel souboru

(zdroj: vlastní)

5.4.2 leaderboard.mat

Soubor **leaderboard.mat** slouží k ukládání výsledků jednotlivých hráčů. Obsahuje proměnnou **Leaderboard**, což je pole struktur se třemi položkami:

- **Jmeno** – jméno hráče zadané po dokončení hry
- **Cas** – celkový čas (v sekundách), za který hráč hru dokončil
- **Chyby** – počet chybných odpovědí hráče

Na obrázku 32 je znázorněno zobrazení datové struktury **leaderboard.mat**.

```
>> load('leaderboard.mat', 'Leaderboard')
disp(struct2table(Leaderboard))
```

Jmeno	Cas	Chyby
"Martin"	22.69	0
"Pavel"	78.746	0
"Ondra"	79.965	1
"David"	70.926	0
"Dominik"	56.795	2
"Vašek"	434.64	6
"Damian"	193.09	2
"Daniel"	77.068	1
"Jan"	66.826	0
"Bedrich"	266.2	4
"Olda"	63.193	0
"Míša"	81.402	1
"Vašek"	67.689	1
"Bory"	76.877	1
"Denisa"	136.15	1
"Neznámý hráč"	254.78	4
"Neznámý hráč"	118.65	1
"Neznámý hráč"	152.57	1
"Neznámý hráč"	70.242	0
"Neznámý hráč"	116.91	0
"Denis"	81.442	1

Obrázek 32 Zobrazení datové struktury leaderboard.mat

(zdroj: vlastní)

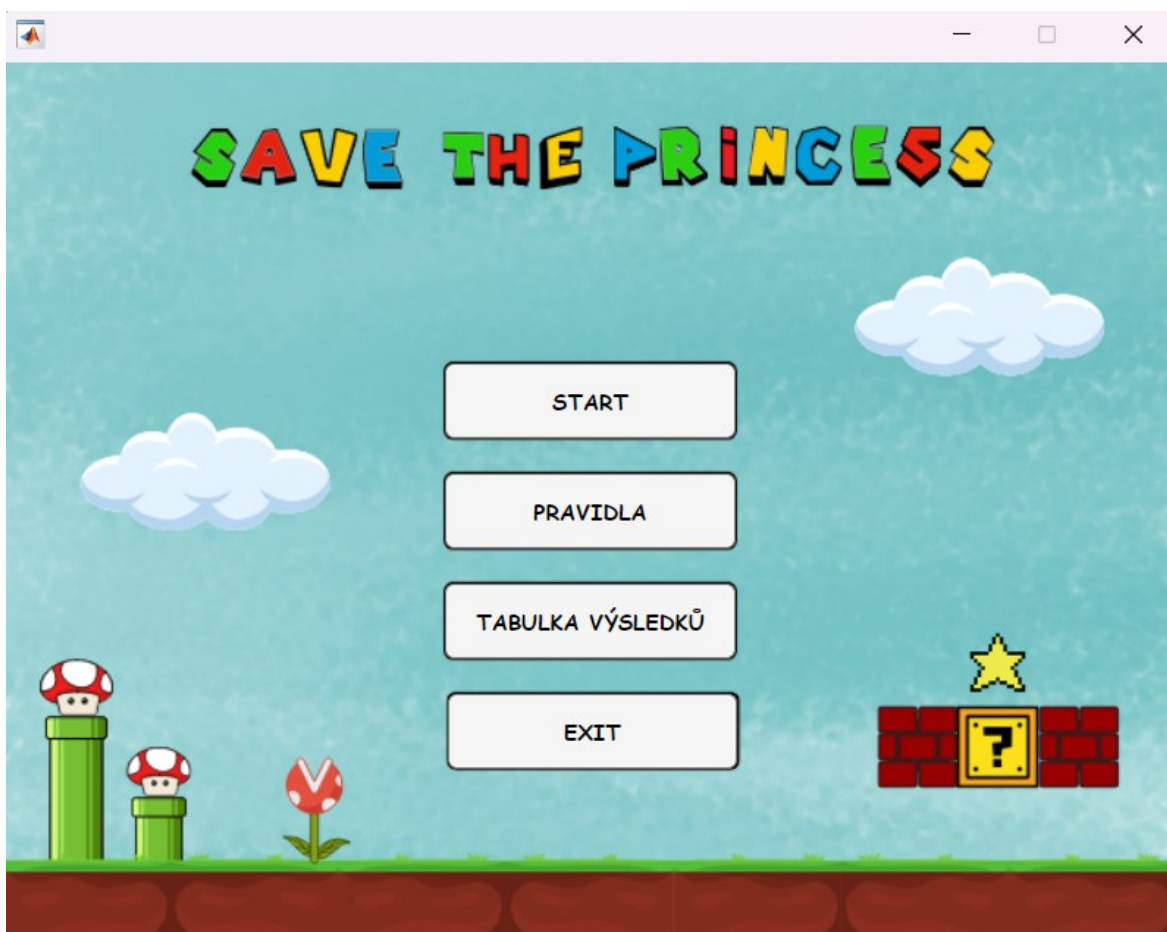
5.5 Pohled ze strany uživatele

5.5.1 Spuštění aplikace

Po spuštění aplikace se zobrazí hlavní menu, které můžeme vidět na obrázku 33.

Uživatel má v hlavním menu na výběr ze čtyř možností:

- **START** – spustí se hra.
- **PRAVIDLA** – otevře nové okno s herními pravidly a popisem ovládání.
- **TABULKA VÝSLEDKŮ** – zobrazí výsledkovou tabulku s uloženými skóre všech hráčů.
- **EXIT** – ukončí aplikaci po potvrzení dialogovým oknem.

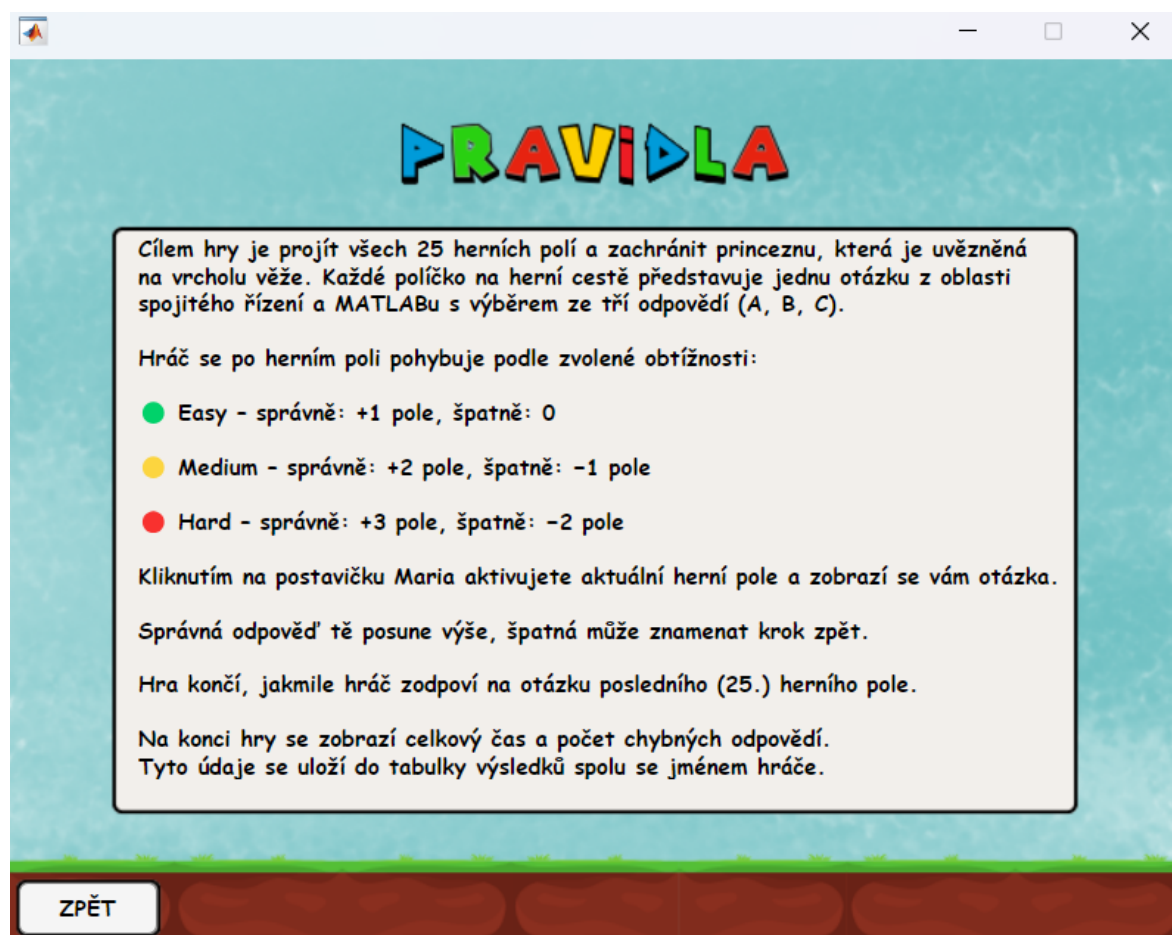


Obrázek 33 Hlavní menu z pohledu uživatele

(zdroj: vlastní)

5.5.2 Zobrazení pravidel

Po kliknutí na tlačítko **PRAVIDLA** se otevře nové přehledné okno, které vysvětluje cíl hry, princip odpovídání na otázky, systém bodování podle obtížnosti i podmínky pro vítězství. Tlačítkem **ZPĚT** se uživatel navrátí do hlavního menu. Toto okno je zobrazeno na obrázku 34.

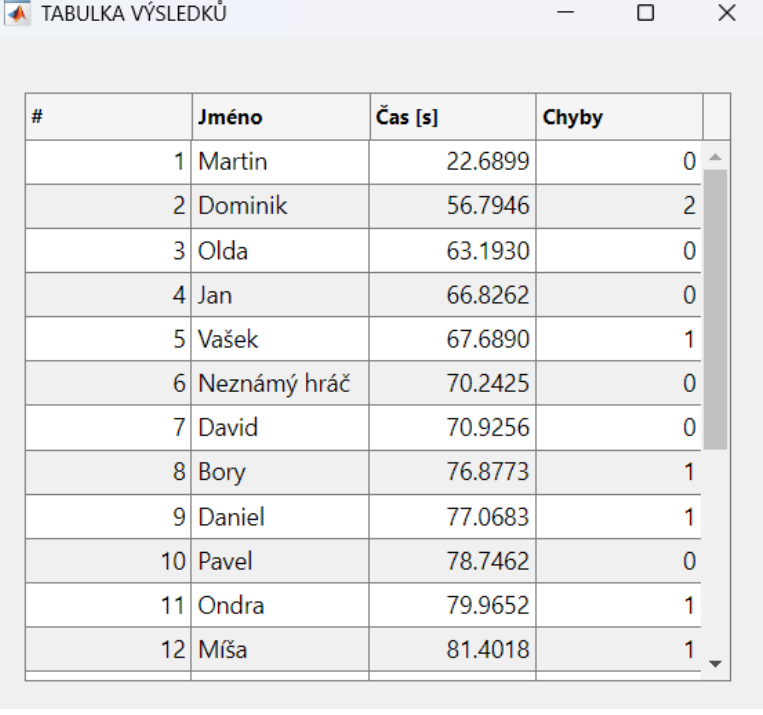


Obrázek 34 Pravidla z pohledu uživatele

(zdroj: vlastní)

5.5.3 Tabulka výsledků

Po kliknutí na tlačítko **TABULKA VÝSLEDKŮ** se zobrazí tabulka s výsledky všech hráčů z předešlých her. Na obrázku 35 je zobrazena tabulka výsledků z pohledu uživatele.



The screenshot shows a window titled "TABULKA VÝSLEDKŮ" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window is a table with four columns: "#", "Jméno", "Čas [s]", and "Chyby". The table contains 12 rows of data, representing different players and their performance. A vertical scrollbar is visible on the right side of the table, indicating that the table can be scrolled.

#	Jméno	Čas [s]	Chyby
1	Martin	22.6899	0
2	Dominik	56.7946	2
3	Olda	63.1930	0
4	Jan	66.8262	0
5	Vašek	67.6890	1
6	Neznámý hráč	70.2425	0
7	David	70.9256	0
8	Bory	76.8773	1
9	Daniel	77.0683	1
10	Pavel	78.7462	0
11	Ondra	79.9652	1
12	Míša	81.4018	1

Obrázek 35 Tabulka výsledků z pohledu uživatele

(zdroj: vlastní)

5.5.4 Zahájení hry

Po kliknutí na **START** se spustí herní okno. Uživatel si v levé části obrazovky vybere požadovanou obtížnost otázky (Easy, Medium, Hard). Zbytek obrazovky tvoří herní pole s 25 políčky a grafickým pozadím. Během hry má uživatel k dispozici tři funkční tlačítka: **RESET** pro resetování hry na začátek, **EXIT** pro zavření celé aplikace a **ZPĚT**, které ho přesměruje zpět do hlavního menu. Obrázek 36 znázorňuje herní okno po spuštění hry z pohledu uživatele.

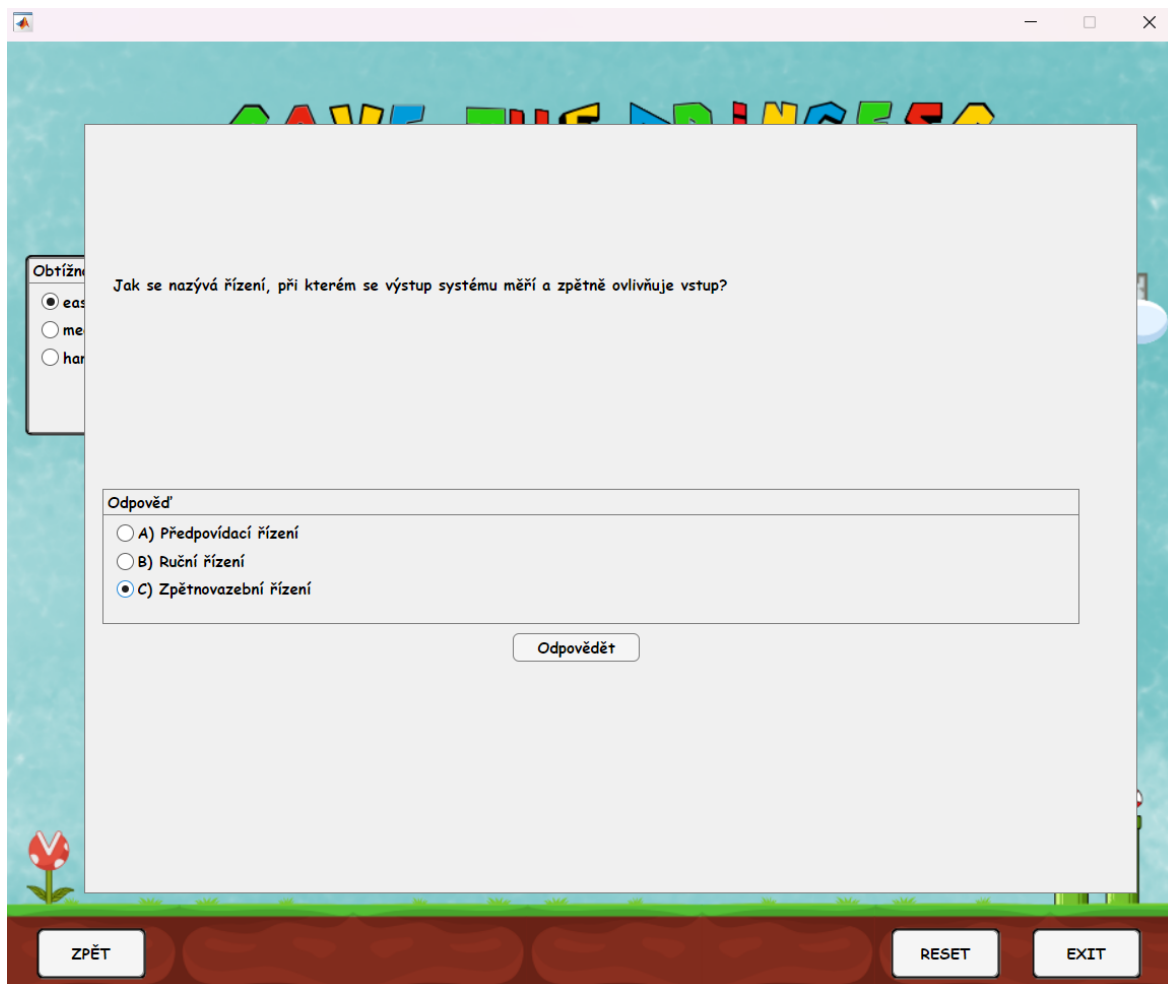


Obrázek 36 Zahájení hry z pohledu uživatele

(zdroj: vlastní)

5.5.5 Odpovídání na otázky

Hráč klikne na políčko, na kterém se nachází postavička Maria. Zobrazí se panel s náhodně vybranou otázkou a třemi možnými odpověďmi (A, B, C). Po výběru odpovědi klikne hráč na tlačítko **Odpovědět**. Tento panel můžeme vidět na obrázku 37.

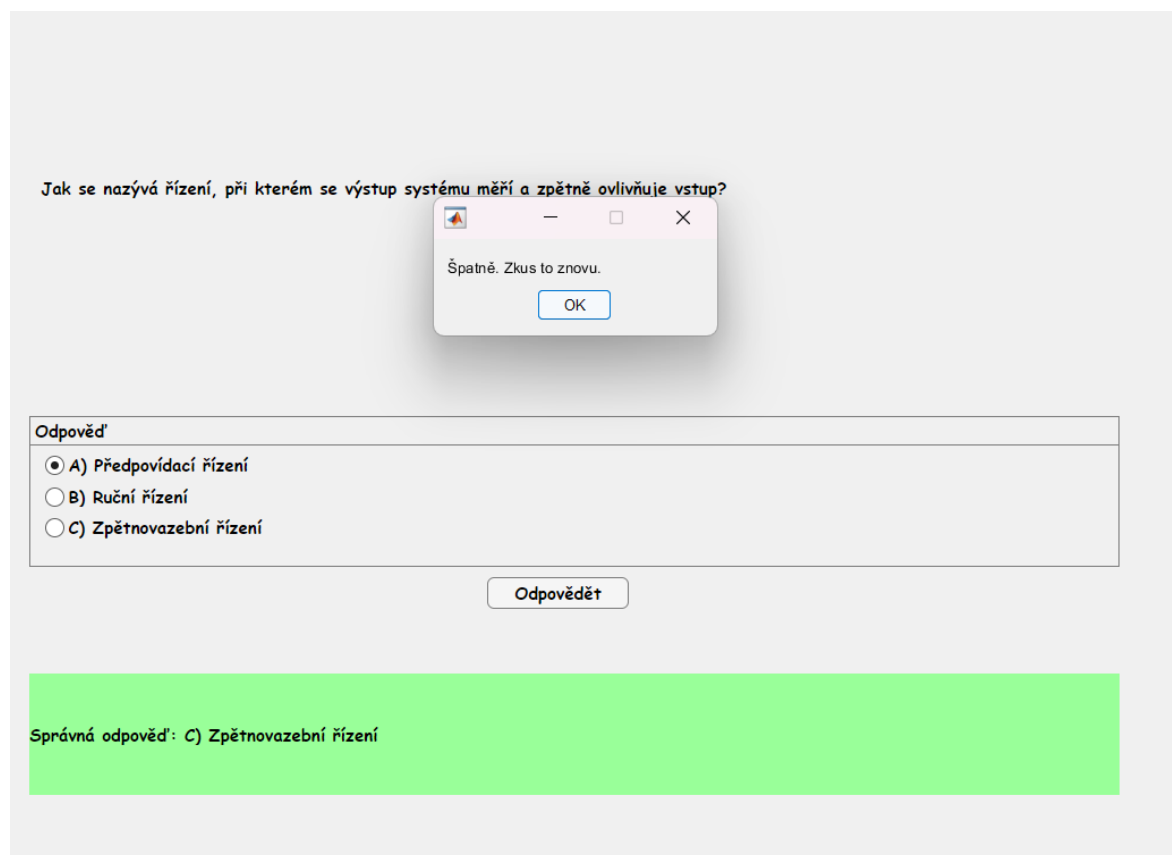


Obrázek 37 Zobrazení otázky z pohledu uživatele

(zdroj: vlastní)

5.5.6 Vyhodnocení odpovědi

Po zodpovězení otázky se hráči zobrazí zpráva, zda odpověděl správně či nikoli. Pokud byla odpověď špatná, zobrazí se navíc i správná odpověď. Hráč je podle správnosti posunut po herním poli vpřed či zpět. Obrázek 38 zachycuje situaci, kdy uživatel odpověděl špatně.



Obrázek 38 Vyhodnocení špatné odpovědi z pohledu uživatele

(zdroj: vlastní)

5.5.7 Postup hry

Správně zodpovězená políčka se po odpovědi zbarví zeleně a stanou se neaktivními, aby na ně již nešlo znovu kliknout. Hráč se v takovém případě posune dopředu o počet polí odpovídající zvolené obtížnosti. V případě nesprávné odpovědi se hráč vrací zpět o počet polí odpovídající zvolené obtížnosti. Na obrázku 39 je zachycen postup hráče ve hře.

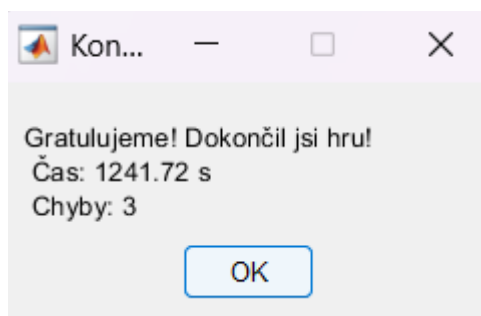


Obrázek 39 Postup hry z pohledu uživatele

(zdroj: vlastní)

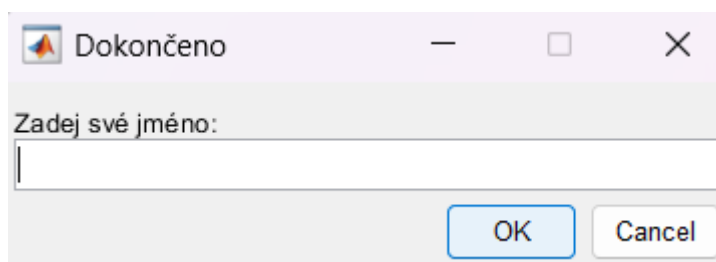
5.5.8 Dokončení hry

Jakmile hráč odpoví na otázku a jeho pozice dosáhne, nebo překročí poslední 25. pole, hra se ukončí. Zobrazí se výsledná zpráva s časem a počtem chyb. Tuto zprávu můžeme vidět na obrázku 40. Následně je hráč vyzván k zadání jména, které se spolu s výsledkem uloží do výsledkové tabulky. Toto dialogové okno můžeme vidět na obrázku 41.



Obrázek 40 Výsledná zpráva z pohledu uživatele

(zdroj: vlastní)

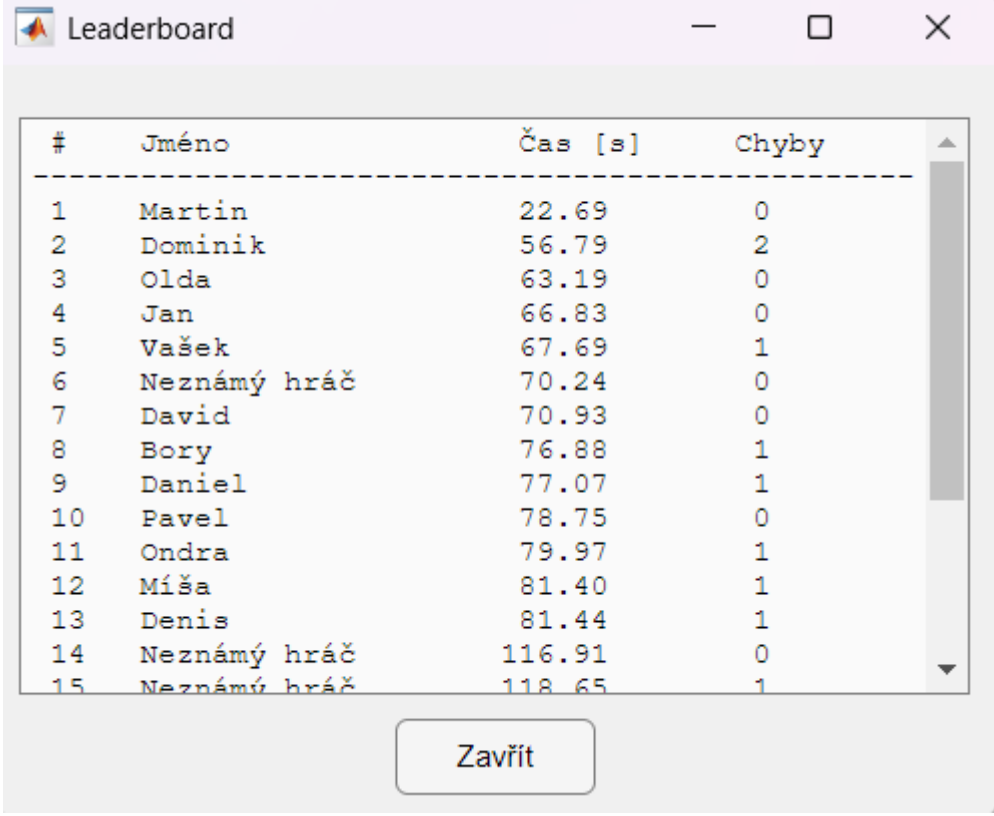


Obrázek 41 Zadání jména hráče z pohledu uživatele

(zdroj: vlastní)

5.5.9 Tabulka výsledků

Po dokončení hry se uživateli zobrazí tabulka výsledků, kterou můžeme vidět na obrázku 42. Ta je seřazena podle dosaženého času a zobrazuje jméno, čas a počet chyb jednotlivých hráčů.



#	Jméno	Čas [s]	Chyby
1	Martin	22.69	0
2	Dominik	56.79	2
3	Olda	63.19	0
4	Jan	66.83	0
5	Vášek	67.69	1
6	Neznámý hráč	70.24	0
7	David	70.93	0
8	Bory	76.88	1
9	Daniel	77.07	1
10	Pavel	78.75	0
11	Ondra	79.97	1
12	Míša	81.40	1
13	Denis	81.44	1
14	Neznámý hráč	116.91	0
15	Neznámý hráč	118.65	1

Obrázek 42 Zobrazení tabulky výsledků z pohledu uživatele

(zdroj: vlastní)

5.6 Testování aplikace

Aplikace byla během vývoje opakovaně testována s cílem ověřit její stabilitu, funkčnost a uživatelskou přívětivost. Testování probíhalo přímo v prostředí MATLAB App Designer v průběhu programování i ve formě výsledné standalone aplikace po exportu do .exe souboru.

Při testování byly ověřeny následující klíčové prvky:

- spuštění a správná funkčnost hlavního menu a přechod mezi jednotlivými částmi aplikace
- správné načtení otázek z excel souboru (**questions.xlsx**)
- náhodné míchání otázek a odpovědí
- posun hráče po herním poli v závislosti na obtížnosti a správnosti odpovědi
- správné ukončení hry a vyhodnocení výsledků
- uložení výsledků do souboru **leaderboard.mat** a jejich zobrazení v tabulce

Aplikace byla rovněž testována několika testery, kteří poskytli zpětnou vazbu z pohledu uživatele. Na základě jejich podnětů byly upraveny některé prvky uživatelského rozhraní. Například doplnění zobrazení správné odpovědi v případě chybné odpovědi.

Během testování se objevily i některé technické problémy:

Náhodné zobrazování otázek – Po spuštění aplikace se vždy zobrazovala stejná první otázka, i když měla být vybrána náhodně. Problém byl v tom, že generátor náhodných čísel generoval vždy stejnou posloupnost. Chyba byla opravena přidáním příkazu **rng('shuffle')**, který zajistí, že se otázky skutečně vybírají náhodně při každém spuštění.

Zásobník a neopakovatelnost otázek – Aby se v jedné hře neopakovala stejná otázka vícekrát, byla vytvořena dvojice zásobníků pomocí dvou oddělených struktur. Zásobník **All**, který obsahuje všechny otázky načtené z Excelu a zásobník **Remaining**, který obsahuje zatím nepoužité otázky, ze kterého se během hry náhodně používají a následně odstraňují. Po použití otázek se zásobník obnoví.

Míchání odpovědí – Jedním z náročnějších úkolů při vývoji aplikace bylo správné naprogramování logiky pro zamíchání odpovědí. Cílem bylo, aby možnosti A, B a C nebyly vždy ve stejném pořadí, ale zároveň bylo nutné zachovat informaci o tom, která z nich je správná i po přeházení. Řešením bylo vytvoření bloku kódu, který náhodně promíchá pořadí odpovědí a zároveň zpětně dohledá, kde se nově nachází ta správná. Tato pozice je poté uložena pro pozdější vyhodnocení odpovědi hráče.

5.7 Zhodnocení aplikace

Aplikace byla navržena tak, aby spojovala hravý vizuální styl s praktickým využitím při učení. Slouží jako interaktivní a zábavná pomůcka pro procvičování zvolené látky. Umožňuje studentům aktivně opakovat a upevňovat své znalosti, přičemž zároveň poskytuje okamžitou zpětnou vazbu o správnosti odpovědí a počtu chyb.

Jednou z hlavních výhod aplikace je, že databáze otázek je uložena ve formátu běžného Excel souboru. Díky tomu si kdokoli může snadno otázky upravit nebo doplnit podle vlastních potřeb. Obsah tak lze jednoduše přizpůsobit výuce libovolného předmětu, nejen tématu spojitého řízení. Aplikace je proto univerzálně využitelná všude tam, kde lze učivo ověřovat formou otázek s výběrem odpovědí.

Studenti si díky intuitivnímu rozhraní a herním prvkům procvičí látku zábavnou formou, což usnadňuje zapamatování informací.

Do budoucna by bylo možné aplikaci dále rozšířit, například o editor otázek přímo v aplikaci, zvukovou odezvu nebo animace.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a realizovat aplikaci pro podporu výuky spojitého řízení v prostředí MATLAB s využitím nástroje App Designer. Výsledkem je funkční a graficky atraktivní aplikace s názvem Save the Princess, která kombinuje prvky kvízu a zábavné mechaniky deskové hry.

Herní plocha je stylizována jako stoupající schodiště s 25 políčky, které vedou k věži, kde je uvězněna princezna. Na těchto polích se pohybuje postavička Maria, jenž vizuálně znázorňuje aktuální pozici hráče. Celé prostředí doplňuje barevné pozadí a tematické grafické prvky, které přispívají k hravé a poutavé atmosféře aplikace. Cílem hry je zodpovídáním otázek dosáhnout vrcholu věže a symbolicky tak zachránit princeznu.

Aplikace umožňuje studentům zábavnou formou utříbit a opakovat své znalosti prostřednictvím odpovědí na otázky, které jsou tematicky zaměřeny na spojitě řízení a práci v prostředí MATLAB. Díky přehlednému grafickému rozhraní a herním prvkům poskytuje interaktivní způsob učení. Kdokoliv si může snadno upravit databázi otázek, která je uložena v excel souboru, a přizpůsobit tak obsah výuce jakéhokoliv předmětu.

V teoretické části práce byly představeny základní principy spojitého řízení a prostředí MATLAB, včetně nástroje App Designer, který slouží pro tvorbu aplikací s grafickým uživatelským rozhraním. Součástí rešerše byla také analýza vybraných aplikací pro podporu výuky, vytvořených ostatními studenty.

Praktická část se následně věnovala návrhu a tvorbě vlastní aplikace, popisu její struktury, použitých funkcí a vyhodnocení její funkčnosti. Výsledná aplikace je připravena k využití v praxi a může být dále rozšiřována dle konkrétních požadavků výuky.

SEZNAM POUŽITÉ LITERATURY

- [1] DUŠEK, František. *MATLAB a SIMULINK: úvod do používání*. Pardubice: Univerzita Pardubice, 2000. ISBN 80-719-4273-1.
- [2] MATHWORKS. *Products and Services*. Online. 2025. Dostupné z: <https://www.mathworks.com/products.html>. [cit. 2025-05-24].
- [3] PERŮTKA, Karel. *MATLAB – základy pro studenty automatizace a informačních technologií*. Učební texty vysokých škol. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. ISBN 80-731-8355-2. Dostupné také z: <http://kramerusndk.nkp.cz/search/handle/uuid:f413fb00-edc1-11e6-bc0e-005056827e51>.
- [4] MATHWORKS. *What Is a Live Script or Function?* Online. 2025. Dostupné z: https://www.mathworks.com/help/matlab/matlab_prog/what-is-a-live-script-or-function.html. [cit. 2025-05-24].
- [5] MATHWORKS. *Conditional Statements*. Online. 2025. Dostupné z: https://www.mathworks.com/help/matlab/matlab_prog/conditional-statements.html. [cit. 2025-05-24].
- [6] MATHWORKS. *Loop Control Statements*. Online. 2025. Dostupné z: https://www.mathworks.com/help/matlab/matlab_prog/loop-control-statements.html. [cit. 2025-05-24].
- [7] MATHWORKS. *Entering Commands*. Online. 2025. Dostupné z: https://www.mathworks.com/help/matlab/entering-commands.html?s_tid=CRUX_topnav. [cit. 2025-05-24].
- [8] MATHWORKS. *2-D and 3-D Plots*. Online. 2025. Dostupné z: https://www.mathworks.com/help/matlab/2-and-3d-plots.html?s_tid=CRUX_lftnav. [cit. 2025-05-24].
- [9] MATHWORKS. *Plotedit*. Online. 2025. Dostupné z: <https://www.mathworks.com/help/matlab/ref/plotedit.html>. [cit. 2025-05-24].
- [10] MATHWORKS. *Develop Apps Using App Designer*. Online. 2025. Dostupné z: <https://www.mathworks.com/help/matlab/app-designer.html>. [cit. 2025-05-24].

- [11] VÍTEČKOVÁ, Miluše a VÍTEČEK, Antonín. *Základy automatické regulace*. 2. vyd. Ostrava: VŠB – Technická univerzita Ostrava, 2019. ISBN 978-80-248-1924-2.
- [12] BALÁTĚ, Jaroslav. *Automatické řízení*. 2., přeprac. vyd. Praha: BEN, 2004. ISBN 80-730-0148-9.
- [13] PROKOP, Roman; MATUŠŮ, Radek a PROKOPOVÁ, Zdenka. *Teorie automatického řízení – lineární spojité dynamické systémy*. Učební texty vysokých škol. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006. ISBN 80-731-8369-2. Dostupné také z: <http://kramerusndk.nkp.cz/search/handle/uuid:3eeb95b0-2fcc-11e7-a38c-005056827e51>.
- [14] NAVRÁTIL, Pavel, 2011. Automatizace: vybrané statě. Ve Zlíně: Univerzita Tomáše Bati ve Zlíně, 289 s. ISBN 978-80-7318-935-8. Dostupné také z: <http://hdl.handle.net/10563/18581>
- [15] OPLUŠTILOVÁ, Petra. *Návrh aplikace a sady příkladů pro podporu výuky automatizace pro prezenční a kombinované studium*. Bakalářská, vedoucí Perůtka, Karel. Zlín: Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav automatizace a řídicí techniky, 2018. Dostupné také z: <http://hdl.handle.net/10563/43165>.
- [16] VYORALOVÁ, Jana. *Podpora cvičení z předmětu: Teorie automatického řízení I*. Bakalářská, vedoucí Navrátil, Pavel. Zlín: Tomas Bata University in Zlín. Faculty of Applied Informatics, Ústav automatizace a řídicí techniky, 2007. Dostupné také z: <http://hdl.handle.net/10563/3129>.
- [17] HAUSEKER, Ladislav. *Grafický nástroj pro podporu výuky teorie systémů a řízení*. Online, Diplomová, vedoucí Michal Řepka. Ostrava: Vysoká škola báňská – Technická univerzita Ostrava, 2013. Dostupné z: <http://hdl.handle.net/10084/98023>.
[cit. 2025-05-25].
- [18] NOVOTNÝ, Matěj. *Výukové animace pro předmět Spojité řízení*. Bakalářská, vedoucí Pekař, Libor. Zlín: Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav automatizace a řídicí techniky, 2022. Dostupné také z: <http://hdl.handle.net/10563/51520>.

- [19] VIČÁNEK, Martin. *Multimediální podpora výuky předmětu TAŘ II – Syntéza*. Bakalářská, vedoucí Vašek, Vladimír. Zlín: Univerzita Tomáše Bati ve Zlíně. Fakulta technologická, Institut řízení procesů a aplikované informatiky, 2005. Dostupné také z: <http://hdl.handle.net/10563/36252>.

SEZNAM OBRÁZKŮ

Obrázek 1 Rozhraní aplikace MATLAB po spuštění	16
Obrázek 2 Interaktivní editace grafu.....	24
Obrázek 3 Úvodní okno App Designeru	25
Obrázek 4 Vývojové prostředí App Designeru	26
Obrázek 5 Code View v App Designeru	27
Obrázek 6 Blokové schéma ovládání.....	28
Obrázek 7 Blokové schéma regulace	29
Obrázek 8 Obecné blokové schéma regulačního obvodu	29
Obrázek 9 Schéma řešení problému pomocí L-transformace	31
Obrázek 10 Přehled a vlastnosti ideálních regulátorů.....	39
Obrázek 11 Přechodová charakteristika proporcionální soustavy 0. řádu	41
Obrázek 12 Přechodová charakteristika proporcionální soustavy 1. řádu	42
Obrázek 13 Přechodová charakteristika proporcionální soustavy 2. řádu	43
Obrázek 14 Přechodová charakteristika integrační soustavy 0. řádu	44
Obrázek 15 Přechodová charakteristika integrační soustavy 1. řádu	45
Obrázek 16 Přechodová charakteristika regulované soustavy s dopravním zpožděním	46
Obrázek 17 Stabilita spojitých systémů (a) a diskrétních systémů (b)	46
Obrázek 18 Porovnání Michajlovovy křivky pro stabilní a nestabilní systém.....	50
Obrázek 19 Porovnání uzavřeného a otevřeného regulačního obvodu pro aplikaci Nyquistova kritéria.....	51
Obrázek 20 Průběhy Nyquistovy křivky otevřeného regulačního obvodu v komplexní rovině	52
Obrázek 21 Ukázka ze hry Kviz s otázkami z automatizace	54
Obrázek 22 Ukázka webových stránek	55
Obrázek 23 Ukázka z výukové aplikace	56
Obrázek 24 Ukázka programu	57
Obrázek 25 Ukázka webových stránek	58
Obrázek 26 Vývojový diagram hlavního menu aplikace	63
Obrázek 27 Úvodní obrazovka aplikace Save the Princess	64
Obrázek 28 Okno s pravidly hry Save the Princess	65
Obrázek 29 Vývojový diagram herní logiky	67
Obrázek 30 Herní okno aplikace	68
Obrázek 31 Struktura otázek v Excel souboru	84
Obrázek 32 Zobrazení datové struktury leaderboard.mat	85

Obrázek 33 Hlavní menu z pohledu uživatele	86
Obrázek 34 Pravidla z pohledu uživatele	87
Obrázek 35 Tabulka výsledků z pohledu uživatele.....	88
Obrázek 36 Zahájení hry z pohledu uživatele.....	89
Obrázek 37 Zobrazení otázky z pohledu uživatele	90
Obrázek 38 Vyhodnocení špatné odpovědi z pohledu uživatele	91
Obrázek 39 Postup hry z pohledu uživatele	92
Obrázek 40 Výsledná zpráva z pohledu uživatele	93
Obrázek 41 Zadání jména hráče z pohledu uživatele.....	93
Obrázek 42 Zobrazení tabulky výsledků z pohledu uživatele	94

SEZNAM ZDROJOVÝCH KÓDŮ

Zdrojový kód 1 Funkce startupFcn	69
Zdrojový kód 2 Funkce centerFigure	70
Zdrojový kód 3 Funkce SetupImage	70
Zdrojový kód 4 Funkce LoadQuestions	71
Zdrojový kód 5 Funkce SetupButtons	72
Zdrojový kód 6 Funkce UpdateMarioPosition	73
Zdrojový kód 7 Funkce MarioClicked	73
Zdrojový kód 8 Funkce FieldClicked – spuštění časovavče	74
Zdrojový kód 9 Funkce FieldClicked – obtížnost a obnova otázek	74
Zdrojový kód 10 Funkce FieldClicked – náhodný výběr otázky	74
Zdrojový kód 11 Funkce FieldClicked – zamíchání odpovědí	75
Zdrojový kód 12 Funkce FieldClicked – přiřazení otázky k radiobuttonům	75
Zdrojový kód 13 Funkce FieldClicked – původní správná odpověď	75
Zdrojový kód 14 Funkce FieldClicked – zjištění nové pozice	76
Zdrojový kód 15 Funkce FieldClicked – uložení dat do panelu	76
Zdrojový kód 16 Funkce OdpovdtButtonPushed	77
Zdrojový kód 17 Funkce CorrectAnswer – zobrazení hlášky a posun hráče vpřed	78
Zdrojový kód 18 Funkce CorrectAnswer – zabarvení políček	78
Zdrojový kód 19 Funkce CorrectAnswer – detekce konce hry	79
Zdrojový kód 20 Funkce CorrectAnswer – záznam výsledku a zobrazení tabulky	81
Zdrojový kód 21 Funkce WrongAnswer	82
Zdrojový kód 22 Funkce RESETButtonPushed	83

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Matlab	Matrix Laboratory
Simulink	Simulační nástroj
SQL	Structured Query Language
GUI	Graphical User Interface
L	operátor přímé L-transformace
L^{-1}	operátor zpětné L-transformace
$F(s)$	obraz spojité funkce
s	komplexní proměnná
t	čas
$u(t)$	vstupní veličina
$y(t)$	výstupní veličina
$U(s)$	obraz vstupní veličiny
$Y(s)$	obraz výstupní veličiny
$G_s(s)$	přenos systému
$G_R(s)$	přenos regulátoru
$h(t)$	přechodová funkce
$g(t)$	impulzní funkce
$G(j\omega)$	frekvenční přenos
j	imaginární jednotka
ω	úhlová frekvence
$A(\omega)$	amplituda
$\varphi(\omega)$	fázový posun
$P(\omega)$	reálná část frekvenčního přenosu
$Q(\omega)$	imaginární část frekvenčního přenosu
r_0	váha proporcionální složky regulátoru

r_{-1}	váha integrační složky regulátoru
r_1	váha derivační složky regulátoru
T_I	integrační časová konstanta regulátoru
T_D	derivační časová konstanta regulátoru
$G_O(s)$	přenos otevřeného regulačního obvodu
$G_W(s)$	přenos uzavřeného regulačního obvodu
k	zesílení soustavy
T	časová konstanta

SEZNAM PŘÍLOH

Příloha P I: prilohy.zip

PŘÍLOHA P I: PŘÍLOHY

prilohy.zip – instalátor aplikace a zdrojový kód