

Návod pro knihovnu Simscape

Ondřej Špaček



OBSAH

ÚVOD.....	4
1 SIMULINK	5
1.1 PROSTŘEDÍ KNIHOVEN V SIMULINK.....	5
1.2 PROSTŘEDÍ MODELU V SIMULINK.....	6
1.3 UKÁZKY ZÁKLADNÍCH BLOKŮ A SIMULACÍ V SIMULINK	7
1.4 BLOKY ZDROJŮ, BLOKY PRO ZOBRAZENÍ VÝSLEDKŮ A SIGNÁLŮ	8
1.4.1 Řešení diferenciálních rovnic v Simulink	9
1.4.2 Řešení soustav diferenciálních rovnic v Simulink	11
1.4.3 Užití přenosové funkce a PID regulátoru v Simulink	12
1.4.4 Použití stavového popisu v Simulink.....	14
1.5 SPRÁVNÁ VOLBA ŘEŠITELE „ <i>SOLVER</i> “	15
2 SIMSCAPE	19
2.1 SEZNÁMENÍ S DŮLEŽITÝMI BLOKY V SIMSCAPE.....	20
2.2 ZÁKLADNÍ KNIHOVNA ELEKTRICKÝCH SOUČÁSTÍ SIMSCAPE	21
2.2.1 Simscape model integračního članku.....	25
2.2.2 Diferenciální rovnice integračního članku	25
2.2.3 Přenos integračního članku	26
2.2.4 Stavový popis integračního članku	27
2.2.5 Porovnání výsledků integračního članku	28
2.2.6 Ukázka řešení složitějšího obvodu v Simscape	29
2.3 ZÁKLADNÍ KNIHOVNA MECHANICKÝCH SOUČÁSTÍ SIMSCAPE	31
2.3.1 Simscape model dvou hmot propojených pružinou	35
2.3.2 Diferenciální rovnice dvou hmot propojených pružinou	36
2.3.3 Stavový popis dvou hmot propojených pružinou.....	37
2.3.4 Přenos dvou hmot propojených pružinou	38
2.3.5 Porovnání výsledků simulací dvou hmot propojených pružinou	40
2.3.6 Složitější mechanický příklad	41
2.4 ZÁKLADNÍ KNIHOVNA HYDRAULICKÝCH SOUČÁSTÍ SIMSCAPE.....	44
2.4.1 Diferenciální rovnice nádrže	47
2.4.2 Simscape model nádrže	48
2.4.3 Linearizace rovnic	51
2.4.4 Přenos linearizované nádrže.....	52
2.4.5 Stavový popis linearizované nádrže	53
2.4.6 Porovnání simulací nádrže	54
2.4.7 Složitější hydraulický příklad	55
3 JAZYK SIMSCAPE.....	58
3.1.1 Nutná adresářová struktura a její překlad.....	58
3.2 SYNTAX JAZYKA SIMSCAPE.....	59
3.2.1 Bloky tříd	59
3.2.2 Seznam bloků kódu pro deklaraci fyzikální oblasti nebo komponenty	60
3.2.3 Vysvětlení jednotlivých bloků deklarace, postup tvorby kódu komponenty	61
3.2.4 Blok MATLAB funkce „ <i>function setup</i> “	65
3.2.5 Blok rovnic „ <i>equations</i> “	68

3.3	FYZIKÁLNÍ OBLAST „DOMAIN“	73
3.3.1	Struktura souboru fyzikální oblasti	73
3.3.2	Globálně nastavené parametry vs. rozlévání parametrů.....	74
3.4	KOMPONENTY „COMPONENT“	75
3.4.1	Vytvoření podtřídy komponenty zděděním vlastností rodičovské třídy	76
3.4.2	Kompozitní komponent, vytvoření instance třídy komponenty.....	77
3.4.3	Směr signálu, polarita portů a referenční bod	81
3.4.4	Obrázek, popisky vstupů/výstupů, název bloku komponenty	82
SEZNAM POUŽITÉ LITERATURY.....		85

ÚVOD

Záměrem tohoto návodu je seznámení čtenáře s tvorbou nových Simscape komponent pomocí „*Simscape language*“ a jejich využití. Současně se snahou ukázat výhody a nevýhody knihovny Simscape na ukázkových příkladech.

Celý návod je rozčleněn do tří částí, kde první se zabývá seznámením se Simulink a jeho prostředím, řešení úloh automatizace a správnou volbou řešitele pro daný Simulink model.

Druhá část se zabývá seznámením s veličinovou analogií a základními knihovnami Simscape, konkrétně se jedná o knihovnu elektrickou, mechanickou a hydraulickou. Seznámením probíhá krátkým popisem knihovny a následným popisem všech dostupných bloků. Pro každou knihovnu v každé fyzikální oblasti je uveden jeden příklad modelu v Simscape a jiných matematických modelů jako je diferenciální rovnice, přenos a stavový popis. Cílem této kapitoly je porozumění výhodám a nevýhodám základní knihovny a hlavně ověření správnosti jejich výsledků.

Třetí část je hlavní částí tohoto návodu a je zaměřena na metody vytváření vlastních bloků a fyzikálních oblastí. Podrobně popisuje všechny možnosti Simscape na krátkých ukázkách kódu.

Poznámka:

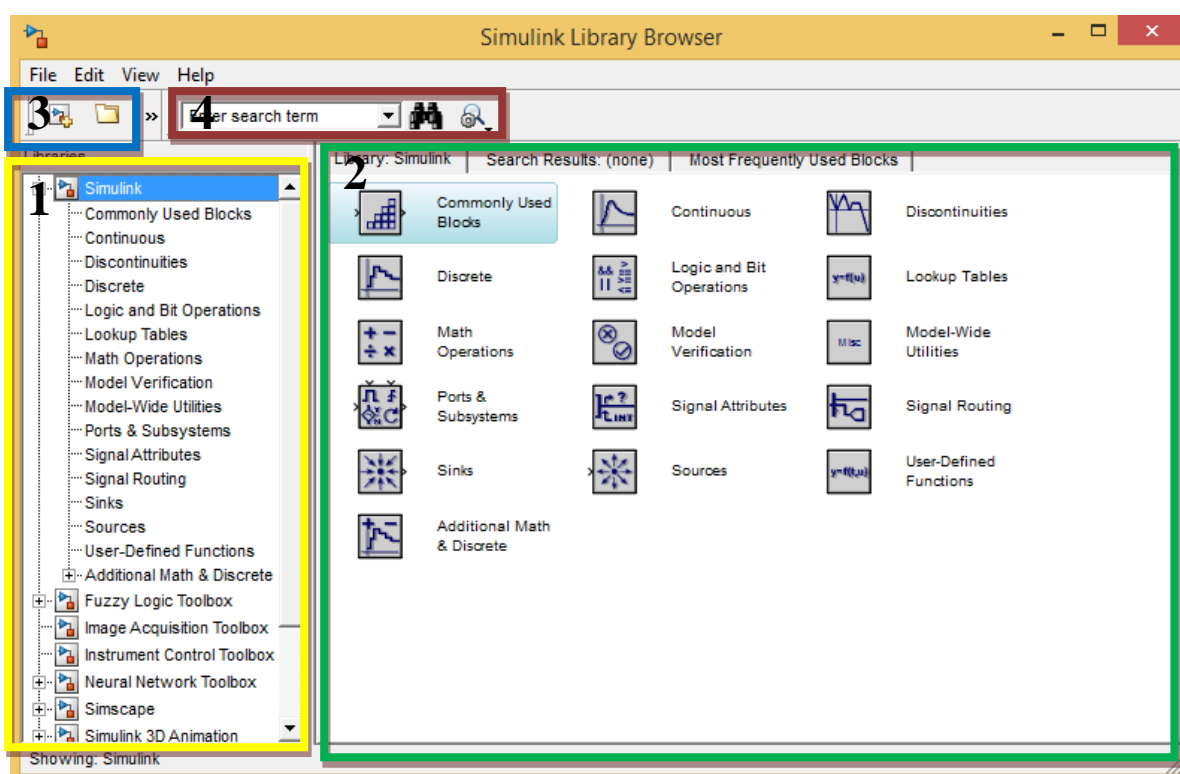
Veškeré uvedené modely byly vytvořeny v rámci bakalářské práce pro MATLAB R2012, kde je postup odvození podrobnější a jsou tam také uvedeny jiné matematické modely ke složitějším systémům.

1 SIMULINK

Jedná se o program v rámci MATLAB, který slouží k simulacím dynamických systémů, kde pro simulace je použito algoritmů MATLAB pro řešení diferenciálních rovnic. Na rozdíl od MATLAB, kde nejčastěji pracujeme s příkazovou řádkou, v Simulink pracujeme interaktivně s blokovým schématem. S pomocí těchto blokových schémat lze vytvořit matematické modely lineárních i nelineárních v čase spojitých nebo diskrétních systémů. Simulink umožňuje tvorbu vlastních uživatelských bloků. Jeho největší výhodou je právě grafická přehlednost grafických schémat, která umožňuje dobrou orientaci i ve složitých matematických modelech. Knihovny Simulink jde nadále rozšiřovat pomocí takzvaného „*blocksetu*“ který obsahuje další bloky a to nejčastěji vytvořených pro určité vědní obory.

1.1 Prostředí knihoven v Simulink

Prostředí knihoven Simulink spustíme pomocí příkazové řádky MATLAB příkazem „*simulink*“, tímto krokem se nám otevře okno „*Simulink Library Browser*“. Následuje popis nejdůležitějších voleb a možností z obrázku tohoto okna.



Obr. 1. Simulink Library Browser

Oblast knihoven „*Libraries*“ na obrázku označena 1, slouží k výběru jednotlivých knihoven, které jsou dále děleny do podkategorií, z důvodu přehlednosti. Počet a typy

knihoven se liší podle různých verzí Simulink, podle přikoupených „*blocksetů*“ a uživatelem vytvořených knihoven.

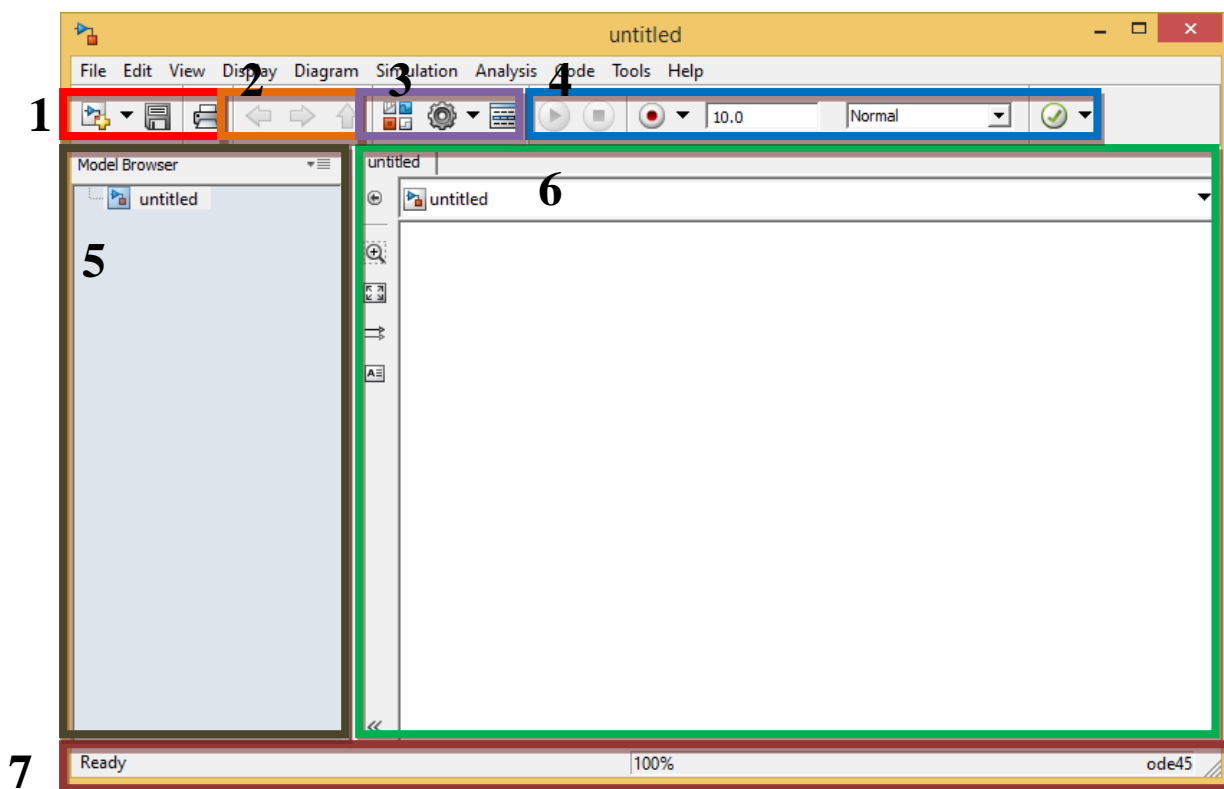
Oblast uvedená na obrázku označena **2**, slouží k procházení jednotlivých bloků knihoven. Je rozdělena na tři záložky v první je aktuálně vybraná knihovna, v druhé bloky nalezené při vyhledávání pomocí funkce „*Search for subsystems and blocks by name*“ a ve třetí se zobrazují nejpoužívanější bloky uživatele Simulink.

Oblast uvedená na obrázku označená **3**, obsahuje dvě ikony první je pro vytvoření Simulink modelu a druhá pro otevření již existujícího Simulink modelu, tyto možnosti najdeme i v nabídce „*File*“.

Oblast pro vyhledávání je na obrázku označená **4**, obsahuje pole pro zadání vyhledávání tlačítko pro vyhledávání „*Search for subsystems and blocks by name*“ a tlačítko pro nastavení ovládání.

1.2 Prostředí modelu v Simulink

Při otevření nového modelu, z „*Simulink Library Browser*“ nebo přímo z MATLAB volbou „*New*“ > „*Simulink Model*“, se zobrazí nové okno s pracovním plátnem. Následuje popis nejdůležitějších funkcí a nastavení z obrázku tohoto okna.



Obr. 2. Pracovní okno Simulink

V oblasti označené **1** na obrázku najdeme tři ikony pro správu modelu, první slouží k vytvoření nového modelu, druhá k uložení současného modelu a třetí k tisku modelu.

V oblasti označené **2** jsou prvky pro krok zpět a krok vpřed třetí ikona se dá použít, pro návrat z podsystému.

V oblasti označené **3** na obrázku najdeme ikonu pro otevření knihovny bloků, ikonu pro nastavení simulace a ikonu pro podrobný průzkum součástí modelu.

V oblasti označené **4** jsou prvky pro ovládání simulace a práci s ní první tlačítko spustí simulaci, druhé tlačítko simulaci zastaví a třetí tlačítko slouží jako jedna z možností logování dat simulací. Také mohou být zobrazeny ikony pro krokování a ikona pro nastavení krokování. Dále je zde pole, ve kterém můžeme rychle měnit délku simulace a nastavení módu simulace. Mód simulace dokáže urychlit simulaci za cenu ztráty některých funkcí a to tak že místo MATLAB kódu použije funkce v jazyku C.

V oblasti označené **5** je prohlížeč subsystémových komponent.

V oblasti označené **6** je plátno, ve kterém se pracuje s bloky, vše funguje na principu „*drag and drop*“, kde jednotlivé bloky berete z knihovny bloků a vkládáte je na plátno přetažením. Jeho součástí je pár ikon jako zoom nebo nastavení na vycentrování modelu do okna a vložení textu.




V oblasti označené **7** je stavový řádek ukazuje nám stav programu potom průběh simulace a zvolený řešitel.

1.3 Ukázky základních bloků a simulací v Simulink


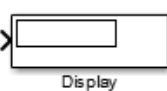

Při řešení jakékoliv úlohy v Simulink potřebujeme určité bloky, které budou vykonávat určitou funkci ve schématu. Všechny základní bloky najdeme v knihovně simulink. Většina bloků po dvou-kliku levého tlačítka myši zobrazí tabulku pro nastavení parametrů bloku a jeho popis. Při nastavování hlavních parametrů bloku např. velikosti signálu můžeme do bloku zadávat přímo číselnou hodnotu nebo proměnnou z MATLAB Workspace. Veškeré bloky dovolují zadávání základních matematických funkcí jako je `sin()`, `cos()`, `exp()` atd. Některé bloky dovolují zadávat i matice, vektory a u některých bloků je matice nebo vektor výchozím parametrem.

1.4 Bloky zdrojů, bloky pro zobrazení výsledků a signálů

Základem je možnost zadávat vstupy a získat výsledky výpočtů k tomu slouží bloky z knihoven „*Sources*“ a „*Sinks*“.

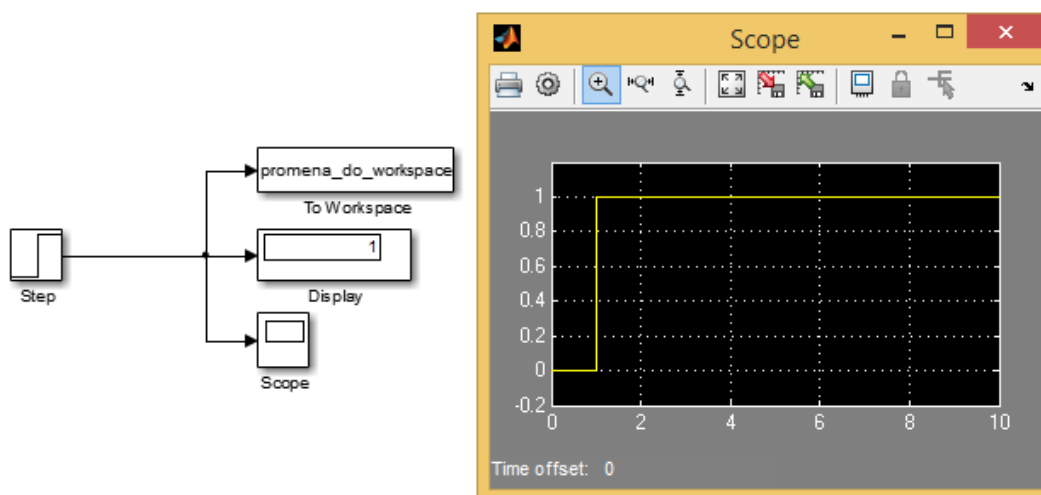
Zdroje signálu a vstupy které nalezneme v knihovně „ <i>Simulink</i> “ > „ <i>Sources</i> “	
Blok	Základní popis
 Constant	<i>Constant</i> má jeden výstup na kterém je konstantní signál, jehož velikost můžeme nastavit v tomto bloku.
 Step	Blok <i>Step</i> má jeden výstup, na kterém je signál v počáteční hodnotě a v určitém čase přejde na jinou hodnotu. Hodnotu signálu a čas skoku můžeme nastavit v tomto bloku.
 Repeating Sequence	Blok <i>Repeating Sequence</i> umožňuje vytvoření vlastního signálu. Který nastavíme v bloku a to dvěma vektory vektorem hodnot a vektorem časů náležících těmto hodnotám.

Tab. 1. Bloky zdrojů Signálu

Bloky pro zobrazení výsledků a signálů ze simulací nalezneme v „ <i>Simulink</i> “ > „ <i>Sinks</i> “	
Blok	Základní popis
 Scope	Blok <i>Scope</i> slouží k zobrazení průběhu signálu v závislosti na čase. Při otevření tohoto bloku se nezobrazí nastavení ale výsledný graf s ovládacími prvky. Jeden vstup odpovídá jednomu plátnu, kde může být i několik grafů dále je možné přidat i další plátna.
 Display	Blok <i>Display</i> slouží k zobrazení jedné hodnoty. V případě více průběžných hodnot se zobrazí hodnota poslední.
 To Workspace	Blok <i>To Workspace</i> nám umožňuje uložení hodnot do prostředí MATLAB kde s daty můžeme provádět další výpočetní operace. V nastavení tohoto bloku je např. volba datové struktury a jméno proměnné, do které bude uložena struktura s daty.

Tab. 2. Bloky pro zobrazení signálů



Na jednoduchém příkladu si můžeme ověřit, že bloky fungují. Vybereme si nějaký blok, který je zdrojem signálu, pro ukázkou je dobré zvolil blok *Step*. Dále si vybereme bloky, které nám zobrazí výsledky. Pro názornost jsem vybral všechny tři zde popisované bloky. Klikneme na šipku vycházející z bloku *Step*, táhneme čáru ke vstupu do bloku *Scope* jakmile šipka zčerná, pustíme a bloky jsou propojeny. Toto provedeme i s bloky *Display* a *To workspace*, kdy signál táhneme vždy s portu bloku k čáře signalizující signál. V bloku *To workspace* můžete ještě změnit jméno proměnné. Spustíme simulaci tlačítkem *Run* a dvojklikem na *Scope* otevřeme výsledný graf. Pro lepší pochopení můžete vyzkoušet různé hodnoty ve *step*, různé zdroje signálu a znovu provést simulaci.


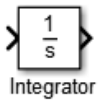


Obr. 3. Ukázka vstup výstup

1.4.1 Řešení diferenciálních rovnic v Simulink

Při řešení základních úloh v teorii systémů řešíme nejčastěji diferenciální rovnice, které lze v Simulink vyřešit pomocí zpětnovazebního obvodu. K řešení diferenciálních rovnic potřebujeme znát následující bloky, které nalezneme v knihovně „Simulink“ > „Commonly Usded Blocks“.

Blok	Základní popis
 	<p>Blok <i>Sum</i> slouží k přičítání a odečítání vstupů kdy na výstupu najdeme výsledek. V nastavení se dá nastavit počet vstupů, jestli se vstup odečítá nebo přičítá pomocí vektoru se znaménky + a -. Dá se nastavit i tvar bloku jako kruh nebo čtverec, který je vhodnější pro velké množství vstupů.</p>

	Blok <i>Gain</i> umožňuje násobení signálu. V zápisu násobící konstanty umožňuje použití základních matematických funkcí MATLAB (např <code>cos()</code> , <code>sqrt()</code> , <code>exp()</code> ...), do násobící konstanty lze zapsat matici i vektor.
	Blok <i>Integrator</i> slouží k integraci vstupu. Umožňuje nastavení počátečních podmínek jak proměnnou tak externím signálem.

Tab. 3. Bloky pro práci s diferenciálními rovnicemi

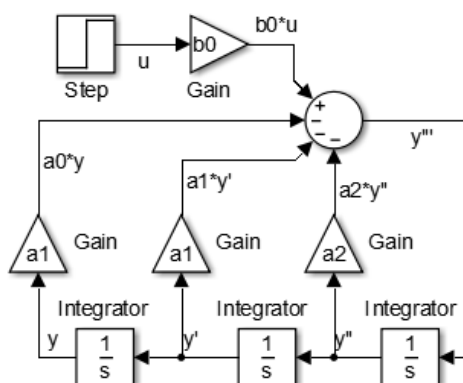
Na ukázkou řešení si zvolíme následující diferenciální rovnici, funkce u , je jednotkový skok a funkce y je neznámá funkce.

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

Nejjednodušší řešení této diferenciální rovnice získáme tak, že si člen s nejvyšší derivací neznámé funkce vyjádříme na jedné straně rovnice a zbytek členů na druhé. Důvodem je to, že neznáme funkci y a vyjádřením členu s nejvyšší derivací nám umožňuje získat řešení jen pomocí integrace v nejjednodušším možném blokovém schématu. Dalším důvodem je, že Simulink se snaží všechny úlohy převést na jednodušší, pokud je to možné a tím může dojít ke zkreslení derivace. Vyjádřením nejvyšší derivace získáme tedy rovnici

$$b_0 u(t) - a_2 y''(t) - a_1 y'(t) - a_0 y(t) = y'''(t)$$

Při pohledu na levou stranu rovnice vidíme, že je zde možné použít blok *Sum*, kdy jeho výstup bude y''' , které budeme postupně integrovat bloky *Integrator* a násobit koeficienty jednotlivých členů pomocí bloku *Gain* a přivádět zpět na blok *Sum*, tak abychom získaly levou stranu rovnice. V bloku *Sum* je třeba nastavit počet vstupů, jestli se budou odečítat, nebo přičítat to vše se dělá po dvojkliku pravým tlačítkem na blok a přidáním znaků $+$ pro vstupy které se budou přičítat a znaků $-$ pro vstupy které se budou odečítat. Jako zdroj použijeme blok *Step*, v případě řešení příkladu s čísly připojíme blok *Scope* k požadovaným větvím schématu. Také je dobré rotovat bloky, tak aby výsledné schéma bylo přehledné, to můžeme učinit klávesovou zkratkou „Ctrl+r“ nebo při kliknutí pravým tlačítkem na blok v menu „*Rotate and Flip*“.



Obr. 4. Ukázka řešení diferenciální rovnice

1.4.2 Řešení soustav diferenciálních rovnic v Simulink

Řešení soustavy diferenciálních rovnic funguje na podobném principu jako u jediné diferenciální rovnice. Soustavu diferenciálních rovnic můžeme řešit například u systémů s více vstupy a výstupy které jsou na sobě vzájemně závislé. Jako příklad jsou zvoleny následující rovnice, kde jsou známy funkce u_1 a u_2 jako skokové funkce a funkce y_1 a y_2 jsou neznámé funkce.

$$y'_1(t) + y_2(t) = u_1(t) - u_2(t)$$

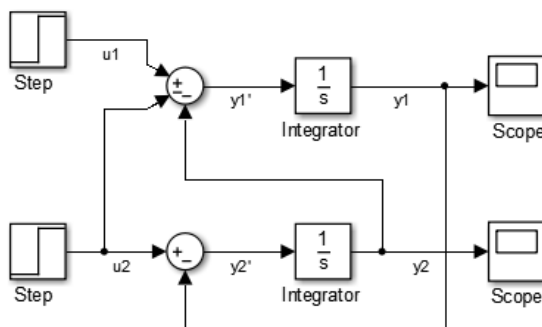
$$y'_2(t) + y_1(t) = u_2(t)$$

Vyjádřením nejvyšší derivace získáme rovnice ve tvaru

$$y'_1(t) = u_1(t) - u_2(t) - y_2(t)$$

$$y'_2(t) = u_2(t) - y_1(t)$$

Tuto soustavu dokážeme vyřešit pomocí dvou zpětnovazebních obvodů, které jsou na sobě závislé a to za použití bloku *Sum*. Jako zdroje signálu jsou použity bloky *Step*, dále jsou použity bloky *Integrator* a *Scope*.




Obr. 5. Ukázka řešení soustavy diferenciálních rovnic

1.4.3 Užití přenosové funkce a PID regulátoru v Simulink

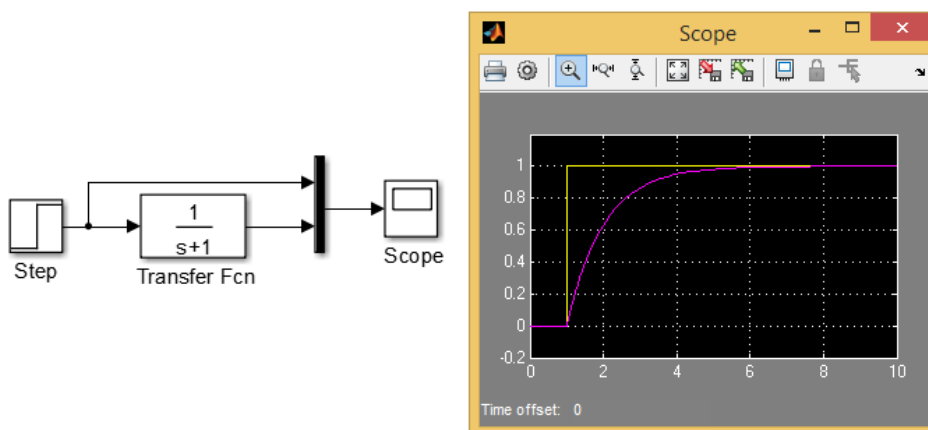
Přenosová funkce je velice často používána jako matematický model systému. V simulaci systému často potřebujeme simulovat nějaké řízení, k tomu se hodně využívá PID regulátor z důvodu jeho jednoduchosti. Pro vytvoření regulačního obvodu budeme potřebovat následující bloky

Blok	Základní popis
 Transfer Fcn	<p>Blok <i>Transfer Fcn</i> umožňuje použití přenosové funkce v obvodu. V jeho nastavení zadáváme koeficienty čitatele a jmenovatele přenosu. Tento blok neumožňuje zadání přenosové matice u vícerozměrných systémů.</p>
 PID Controller	<p>Blok <i>PID Controller</i> slouží jako PID regulátor s nastavitelnými parametry Proporcionální, Integrační a Derivační složky. Tento regulátor je v Simulink popsán následujícím přenosem</p> $PID(s) = P + I \frac{1}{s} + D \frac{N}{1 + N \frac{1}{s}}$
 Mux	<p>Blok <i>Mux</i> slouží jako multiplexor, který z jednotlivých jeho vstupů vytvoří vektor dat. Toho se dá využít například u bloku Scope kdy chceme více průběhů v jednom plátně pro jejich porovnání. V nastavení tohoto bloku můžeme nastavit počet vstupů.</p>

	Blok <i>Demux</i> slouží jako demultiplexor, který vektor dat rozdělí na jednotlivé prvky. Je opakem bloku <i>Mux</i> .
---	---

Tab. 4. Bloky pro regulaci

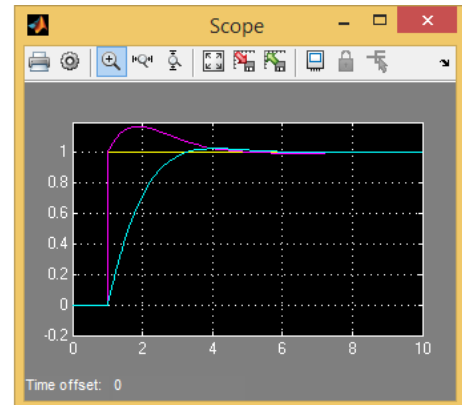
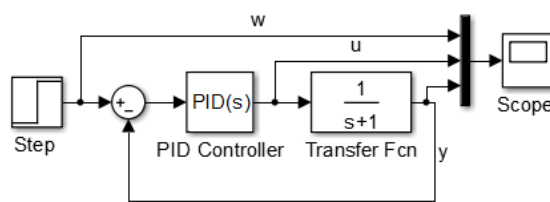
Základním zapojení s přenosem bude zjištění odezvy na určitý signál, proto si vezmeme bloky *Step*, *Transfer Fcn*, *Scope* a propojíme je. Použijeme i bloku *Mux* aby, jsme mohli porovnat vstup a výstup bloku *Transfer Fcn* na jednom plátně. Pro lepší pochopení můžeme měnit hodnoty čitatele a jmenovatele přenosu v bloku *Transfer Fcn*, nebo měnit hodnoty v bloku *Step* a pozorovat výsledky v grafu.



Obr. 6. Ukázka zapojení přenosové funkce

Na obrázku je zobrazeno zapojení a průběh vykreslený pomocí bloku *Scope*, kde žlutou barvou je označen vstup do bloku *Transfer Fcn* a fialovou barvou po průchodu tímto blokem.

Dalším častým zapojením je zpětnovazební zapojení s regulátorem PID. Toto zapojení nám bude simulovat regulaci PID regulátorem na systému, který je popsán přenosem. K zapojení potřebujeme bloky *Step*, *Transfer Fcn*, *PID Controler*, *Sum* a *Scope*. Znovu využijeme bloku *Mux*, pro sledování důležitých veličin v regulačním obvodu a nezapomeneme změnit znaménko v *Sum* na minus.

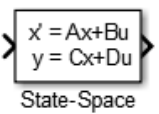


Obr. 7. Ukázka zpětnovazebního regulačního obvodu s PID

Na obrázku je zobrazeno zapojení a průběh regulačního pochodu vykreslený pomocí bloku *Scope*, kde žlutou barvou je označena žádaná veličina w , akční veličina u , je fialovou barvou a regulovaná veličina y je modrou barvou. Simulace je provedena s parametry regulátoru $P=1$, $I=1,5$ a $D=0$. Na obrázku je vidět že průběh signálu za blokem *Transfer Fcn* je rychlejší, než bez PID regulátoru to je způsobeno integrační konstantou PID regulátoru. Pro lepší pochopení je dobré vyzkoušet si změny jednotlivých parametrů PID regulátoru a sledovat změny na bloku *Scope*.

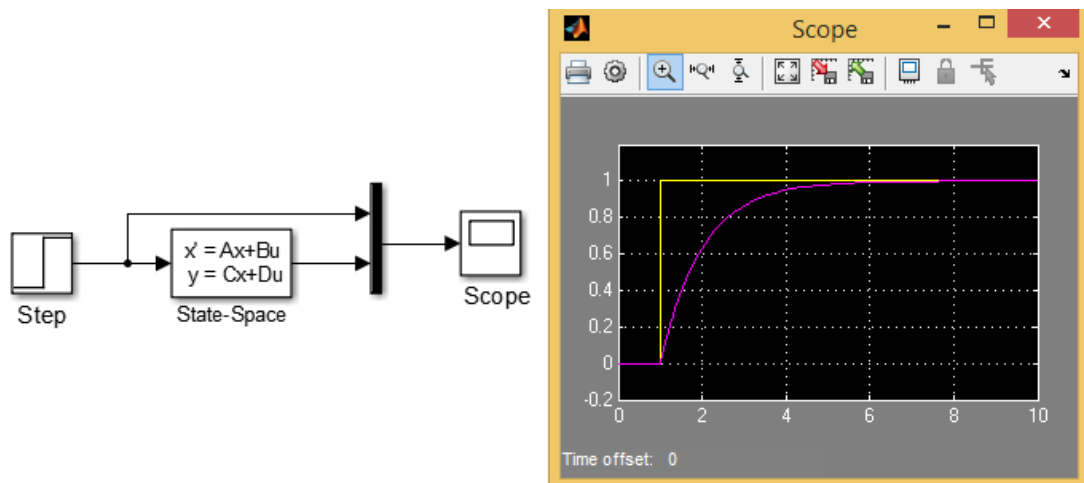
1.4.4 Použití stavového popisu v Simulink

Mezi další úkony v Simulink může být práce se stavovým popisem. Stavový popis je v Simulink vhodné použít hlavně u systémů s více vstupy a výstupy, protože to takto lze udělat jen jedním blokem *State-Space* tento blok nalezneme v knihovně „*Simulink*“ > „*Continuous*“.

Blok	Základní popis
 State-Space	Blok <i>State-Space</i> umožňuje zadání matic stavového popisu A , B , C a D . Tento blok po zadání matic funguje jako matematický model. Tento blok je možné použít i pro popis systémů s více vstupy, výstupy a to připojením multiplexoru ke vstupu a demultiplexoru k výstupu.

Tab. 5. Blok pro stavový popis

Pro ukázkou použijeme blok *Step*, *State-Space*, *Scope* a *Mux*. Do bloku *State-Space* zadáme tyto matice pro rozumný výsledek $A = [-2, -1; 1, 0]$; $B = [1; 0]$; $C = [1, 1]$; $D = [0]$.



Obr. 8. Ukázka bloku stavového popisu

Na obrázku je základní zapojení s blokem *State-Space* na výsledném průběhu vidíme žlutou barvou vstup do bloku *State-Space* a fialovou výstup.

Pro řešení obvodů MIMO se využije bloků *Mux* a *Demux*, připojených na vstup a na výstup. Důležité je do nich zapojit signály ve správném pořadí, tak aby odpovídaly správným řádkům/sloupcům matic.

1.5 Volba řešitele „solver“

Řešitel je algoritmus, který dokáže vyřešit obyčejnou diferenciální rovnici. Řešitelů implementovaných v MATLAB existuje velké množství, Simulink jich využívá jen několik. Každý řešitel využívá jiných metod a přijímá různé parametry jak v MATLAB, tak i v Simulink. Výběrem řešitele rozhodujeme o přesnosti, rychlosti a někdy dokonce i o správnosti řešení. Tyto řešitele mají následující dělení, dle [1].

Podle délky kroku:

- „*Variable-step*“ – Jedná se o metodu proměnného kroku. Jejím principem je, že se zrychlující změnou derivace se zkracuje délka kroku pro dosažení lepší aproximace, v případě že se změna derivace zpomaluje, délka kroku se prodlužuje. Mezi nejpoužívanější řešitele tohoto typu patří ode45, který je i defaultně nastaven při vytvoření nového Simulink schématu.
- „*Fixed-step*“ – Jedná se o metodu neměnného kroku. Je to velice jednoduchá metoda, kdy se výpočty provádí vždy v daném neměnném kroku.

Podle složitosti problému:

- „*Stiff*“ – Obyčejné řešitele často nedokážou vyřešit problémy označované jako stiff. Tyto problémy nastanou, když změna derivace je velice pomalá a v určitém okamžiku se začne velice rapidně měnit. Z toho vyplývá, že k řešení těchto problémů lze použít jen metody s proměnným krokem, ovšem i u některých metod proměnného kroku dojde k chybě a výsledek bude nesprávný. Z tohoto důvodu existují speciální řešitele přímo pro tyto problémy, všechny řešitele vhodné pro tyto problémy mají v Simulink v závorce napsáno „*stiff*“. Nejčastěji používaným řešitelem tohoto typu je řešitel ode15s který je zároveň označován jako nejvhodnější pro fyzikální systémy, tedy i Simscape.
- „*NonStif*“ – Tyto problémy jsou všechny problémy, kde se derivace mění průběžně a ne velice rapidně. Nejběžnějším řešitelem těchto problémů je ode45.

Podle průběhu řešeného problému:

- Diskrétní – Pro řešení diskrétních problémů existuje v Simulink jen jeden řešitel jeho název je „*discrete*“. V případě že Simulink narazí na diskrétní problém, zvolí tento řešitel automaticky.
- Analogový – Většinu problémů řešených v Simulink řešíme v oblasti kdy je průběh analogový. A všechny řešitele obsahující písmena ODE (ordinary differential equation) jsou schopna řešit tyto problémy.

Řešitele Simulink v tabulkách podle přesnosti názvu a použití.

Řešitele neměnného kroku:

Řešitel	Metoda	Řád přesnost
ode1	Euler	první
ode2	Heun	druhý
ode3	Bogacki-Shampin	třetí
ode4	Runge-Kuta (RK4)	čtvrtý
ode5	Dormad-Price (RK5)	pátý
ode8	Dormad-Price (RK8)	osmý

Tab. 6. Řešitele neměnného kroku

Význam řádu přesnosti u metod neměnného kroku je, že čím vyšší řád, tím je metoda přesnější a zároveň je metoda složitější, tedy trvá déle její provedení.

Řešitele proměnného kroku pro problémy „*nonstiff*“:

Řešitel	Metoda	Přesnost	Použití
ode45	Dorman-Price, Runge-Kunta	střední	Tento řešitel je vhodné zvolit jako první. Pokud je řešení touto metodou pomalé možná se jedná o „ <i>stiff</i> “ problém. Je rychlejší a přesnější než ode23.
ode23	Runge-Kunta, Bogacki, Shampine	malá	Tento řešitel je efektivnější než ode45 pro řešení problémů, které mají rychlejší průběh změny derivace.
ode133	PECE, Adams- Bashforth- Moulton	Proměnná (malá až velká)	Vhodný pro řešení za přísných tolerančních podmínek. Tento řešitel může být efektivnější pro řešení problémů s velkým počtem výpočtů, než ode45.

Tab. 7. Řešitele proměnného kroku „*nonstiff*“ problémů

Řešitele proměnného kroku pro „*stiff*“ problémy

Řešitel	Metoda	Přesnost	Použití
ode15s	NDF	Proměnná (malá až střední)	Založen na numerické derivaci místo zpětné, což zvyšuje efektivitu u „ <i>stiff</i> “ problémů. Tento řešitel by měl být první volbou, když zjistíme, že je problém „ <i>stiff</i> “. Řešení jacobiho matice.
ode23s	Modifikace Rosenbrock, druhý řád	malá	Tento řešitel může být efektivnější než ode15s v případě nastavení velmi hrubých tolerancí. Navíc dokáže vyřešit některé „ <i>stiff</i> “ problémy které nedokáže vyřešit ode15s. Řešení jacobiho matice.

ode23t	Trapezoidal, volná interpolace	malá	Tento řešitel je vhodný pro střední „ <i>stiff</i> “ problémy (ne úplně složité). Podmínkou je že nemůže řešit problémy, kde dochází k disipaci energie.
ode23tb	TR-BDF2	malá	Vhodný pro velmi hrubé tolerance. Řešení iterací matice.

Tab. 8. Řešitele proměnného kroku „*stiff*“ problémů

Význam přesnosti řešitelů s proměnným krokem je pouze orientační protože primárně závisí na nastavení tolerancí, řádu použitým v řešení, druh řešitele jacobiho matice v nastavení řešitele.

Kdy nastane čas na změnu řešitele?

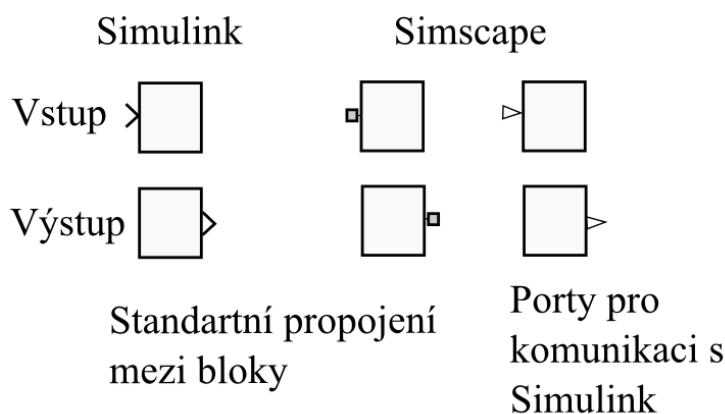
- Když je výsledek jednoznačně špatně, dle kvalifikovaného odhadu (např. výpočet na papíře, experimentálně získané hodnoty).
- Když je výsledkem velice rychle kmitající křivka, nebo vypadá jako „pila“.
- Když řešení trvá příliš dlouho, je možné že problém je „*stiff*“ a je tedy dobré zvolit vhodný řešitel.
- Když řešení nebylo nalezeno v celé oblasti požadovaného intervalu.

2 SIMSCAPE

Simscape je rozšiřující knihovna Simulink. Tato knihovna umožňuje vytvoření matematického modelu v určité konkrétní doméně např. mechanické, elektrické, hydraulické. Jednotlivé bloky v knihovně Simscape odpovídají reálným součástem např. blok pružiny nebo blok rezistoru. Na rozdíl od standartního blokového schématu v Simulink zavádí do schémat fyzikální veličiny a pokročilé nastavení řešitelů diferenciálních rovnic. Z důvodu že bloky reprezentují reálné součásti a signály mezi nimi, jsou signály fyzikálních veličin propojení bloků, tedy veškerá propojení představují přenos energie. Z toho vyplývá největší výhoda knihovny Simscape a to je že nemusíme znát žádné fyzikální zákony, nemusíme získávat matematicko-fyzikální model, protože ze znalosti struktury systému dokážeme vytvořit schéma v Simscape, které bude reprezentovat systém jako jeho matematický model.

Knihovna Simscape má mnoho podknihoven pro konkrétní obory např. SimMechanics, SimHydraulics, SimElectronics, SimDriveline a SimPowerSystems. V případě, že požadovaný blok neexistuje v žádné knihovně Simscape může si ho uživatel vytvořit pomocí programovacího jazyku speciálního pro Simscape „*Simscape language*“.

Zavedení signálů, které odpovídali reálným fyzikálním veličinám, mělo za následek přidání různých druhů portů pro knihovnu Simscape, tak aby nešlo propojit bloky bez jakékoliv souvislosti, například blok *Step* nejde připojit k Simscape bloku rezistoru protože blok *Step* není zdroj elektrické energie. Na následujícím obrázku jsou zobrazeny porty Simulink a porty základní knihovny v Simscape. U všech propojovaných bloků platí, že typ portu výstupu musí být stejný jako vstup dalšího bloku.



Obr. 9. Porty Simulink a Simscape

Rozšiřující podknihovny Simscape mají velké množství dalších portů, specifických pro rozšiřující knihovny.

Při práci se simulovanými reálnými veličinami je třeba si uvědomit, kde a jak je správně změřit. Například v elektrických systémech měříme napětí proti referenci nebo jako rozdíl dvou potenciálů v obvodu a proud jako náboj protékající určitou částí obvodu za čas. Tedy Voltmetr připojíme vždy paralelně k prvku, na kterém měříme proud a ampérmetr sériově. Toto platí u všech domén, jak u reálných systémů, tak v Simscape. Pro pochopení použijeme veličinovou analogii pro domény v základní knihovně, aby bylo jasné jak připojit sensory. Veličiny v obvodě rozdělíme do dvou skupin, ve skupině tok jsou veličiny, které reprezentují rychlost změny za časový okamžik a měříme je tedy sériově k měřenému prvku a ve skupině úsilí jsou prvky, které mají schopnost konat práci a měříme je paralelně k měřenému prvku jako rozdíl dvou potenciálů, dle [2].




Simscape knihovna/Systém	Úsilí [Jednotka]	Tok [Jednotka]
„ <i>Electrical</i> “/Elektrický	Napětí [V]	Proud [A=C/s]
„ <i>Hydraulic</i> “/Hydraulický	Tlak [Pa]	Průtok [m ³ /s]
„ <i>Mechanical</i> “/Mechanický	Síla [N]	Rychlost [m/s]
„ <i>Termal</i> “/Teplený	Teplota [°C]	Tepelný tok [W=J/s]

Tab. 9. Tabulka analogie veličin

Tento návod je věnován nejčastějším příkladovým systémům tedy hydraulickým, elektrickým a mechanickým.

2.1 Seznámení s důležitými bloky v Simscape




Knihovna Simscape využívá speciálních bloků, některé z nich musí být v každém Simscape modelu jako je blok *Solver Configuration*, který slouží pro podrobnější nastavení řešitele diferenciálních rovnic. Dalším blokem, který musí být v každém schématu, je referenční blok, který je specifický pro každou fyzikální doménu. Dalšími důležitými bloky jsou *PS-Simulink Converter* a *Simulink-PS Converter*, tyto bloky slouží jako propojení mezi Simscape a Simulink, slouží pro převod fyzikálních signálů na signály používané v Simulink a naopak, dle [5].




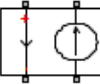
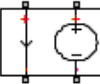
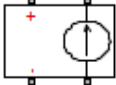
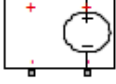
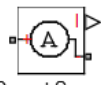

Simscape blok	Základní popis
 Solver Configuration	Blok <i>Solver Configuration</i> slouží k lokálnímu nastavení řešitele diferenciálních rovnic. Standardně je nastavený tak aby používal řešitel nastavený v Simulinku a může být připojen kdekoliv v Simscape obvodu.
 PS-Simulink Converter	Blok <i>PS-Simulink Converter</i> slouží k převodu simulovaného fyzikálního signálu Simscape na signál použitelný v Simulink. Tento blok umožňuje nastavení jednotky signálu.
 Simulink-PS Converter	Blok <i>Simulink-PS Converter</i> . slouží k převodu signálu ze Simulink na signál simulující fyzickou veličinu pro Simscape. V tomto bloku je možné nastavení jednotky a další vlastnosti vstupu.

Tab. 10. Důležité základní bloky Simscape

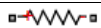




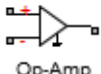

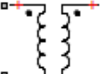


2.2 Základní knihovna elektrických součástí Simscape

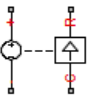
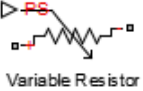


Knihovna Simscape obsahuje mnoho součástí, které jsou používány v elektrických obvodech. Obsahuje prvky měnící elektrickou energii na mechanickou a velké množství různých zdrojů napětí a proudu. Mezi snímače které jsou k dispozici v Simscape pro elektrické obvody patří voltmetr a ampérmetr. Většina z těchto součástí je implementována jako ideální součásti (jinak je to u nich uvedeno).

Zdroje a sensory elektrické fyzikální oblasti	
Simscape blok	Základní popis
 AC Voltage Source  DC Voltage Source  Controlled Voltage Source	Mezi bloky zdroje napětí patří <ul style="list-style-type: none"> • <i>AC Voltage Source</i> je zdroj sinusového střídavého napětí podle zadaného parametru. • <i>DC Voltage Source</i> je zdroj konstantního stejnosměrného napětí. • <i>Controlled Voltage Source</i> je zdroj napětí, jehož napětí je kontrolováno signálem ze Simulink.

 AC Current Source  DC Current Source  Controlled Current Source	<p>Mezi bloky zdroje proudu patří</p> <ul style="list-style-type: none"> • <i>AC Current Source</i> je zdroj sinusového střídavého proudu podle zadaného parametru. • <i>DC Current Source</i> je zdroj konstantního stejnosměrného Proudů. • <i>Controlled Voltage Source</i> je zdroj proudu, jehož proud je kontrolováno signálem ze Simulink.
 Current-Controlled Current Source  Current-Controlled Voltage Source	<p>Mezi zdroje řízené proudem patří</p> <ul style="list-style-type: none"> • <i>Current-Controlled Current Source</i> je blok řízený proudem, který podle nastavitelné konstanty zisku řídí zdroj proudu. • <i>Current-Controlled VoltageSource</i> je blok řízený proudem, který podle nastavitelné konstanty zisku řídí zdroj napětí.
 Voltage-Controlled Current Source  Voltage-Controlled Voltage Source	<p>Mezi zdroje řízené napětím patří</p> <ul style="list-style-type: none"> • <i>Voltage-Controlled Current Source</i> je blok řízený napětím, který podle nastavitelné konstanty zisku řídí zdroj proudu. • <i>Voltage-Controlled VoltageSource</i> je blok řízený napětím, který podle nastavitelné konstanty zisku řídí zdroj napětí.
 Current Sensor	<p>Blok <i>Current Sensor</i> slouží jako ampérmetr s jedním výstupem proudu, který je možné převést do Simulink.</p>
 Voltage Sensor	<p>Blok <i>Voltage Source</i> slouží jako voltmetr s jedním výstupem napětí, který je možné převést na signál Simulink.</p>

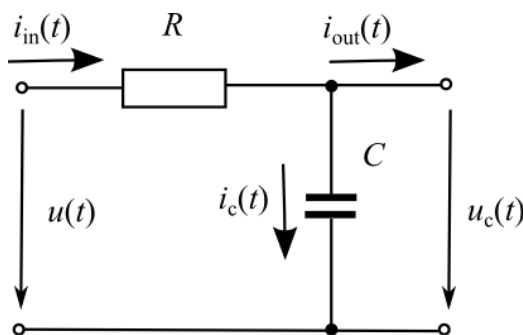
Tab. 11. Bloky zdrojů a sensorů elektrické fyzikální oblasti

Simscape blok	Základní popis
 Resistor	Blok <i>Resistor</i> slouží v obvodu jako rezistor s nastavitelným odporem.
 Capacitor	Blok <i>Capacitor</i> slouží v obvodu jako kondenzátor, u kterého je možné nastavit kapacitu, počáteční napětí a jeho sériovou a paralelní vodivost.
 Inductor	Blok <i>Inductor</i> slouží v obvodu jako induktor, umožňuje nastavení indukčnosti, počátečního proudu a sériovou a paralelní vodivost.
 Electrical Reference	Blok <i>Elektrical Reference</i> slouží jako uzemnění a musí být v každém obvodu minimálně jednou.
 Diode	Blok <i>Diode</i> reprezentuje diodu, s nastavitelným parametrem propustného napětí a odporu.
 Op-Amp	Blok <i>Op-Amp</i> představuje ideální operační zesilovač.
 Gyrator	Blok <i>Gyrator</i> symbolizuje ideální součást gyrator, převádí kapacitní reaktanci na induktivní reaktanci.
 Ideal Transformer	Blok <i>Ideal Transformer</i> reprezentuje ideální transformátor s nastavitelnou konstantou poměru počtu navinutí vodiče.
 Mutual Inductor	Blok <i>Mutual Inductor</i> slouží pro vytvoření vazby mezi dvěma cívkami, nastavitelným parametrem je indukčnost a počáteční proud obou cívek.
 Rotational Electromechanical Converter	Blok <i>Rotational Electromechanical Converter</i> slouží pro převod elektrické energie na rotační mechanickou energii nebo naopak. Zadává se konstanta proporce mezi napětím a rychlostí otáček.

 <p>Translational Electromechanical Converter</p>	<p>Blok <i>Translational Electromechanical Converter</i> slouží pro převod elektrické energie na translační mechanickou energii nebo naopak. Zadává se konstanta poměru mezi napětím a rychlostí posunu.</p>
 <p>Variable Resistor</p>	<p>Blok <i>Variable Resistor</i> umožňuje přivedení signálu Simulink pro řízení odporu rezistoru. Parametrem tohoto bloku je minimální odpor</p>
 <p>Open Circuit</p>	<p>Blok <i>Open Circuit</i> lze použít jako zakončení nepoužitých svorek elektrických svorek (nejedná se o uzemnění)</p>
 <p>Switch</p>	<p>Blok <i>Switch</i> je blok přepínače ovládaný signálem ze Simulink. Parametry je hranice signálu pro otevření, a odpor při zavření/otevření.</p>

Tab. 12. Součásti elektrické fyzikální oblasti

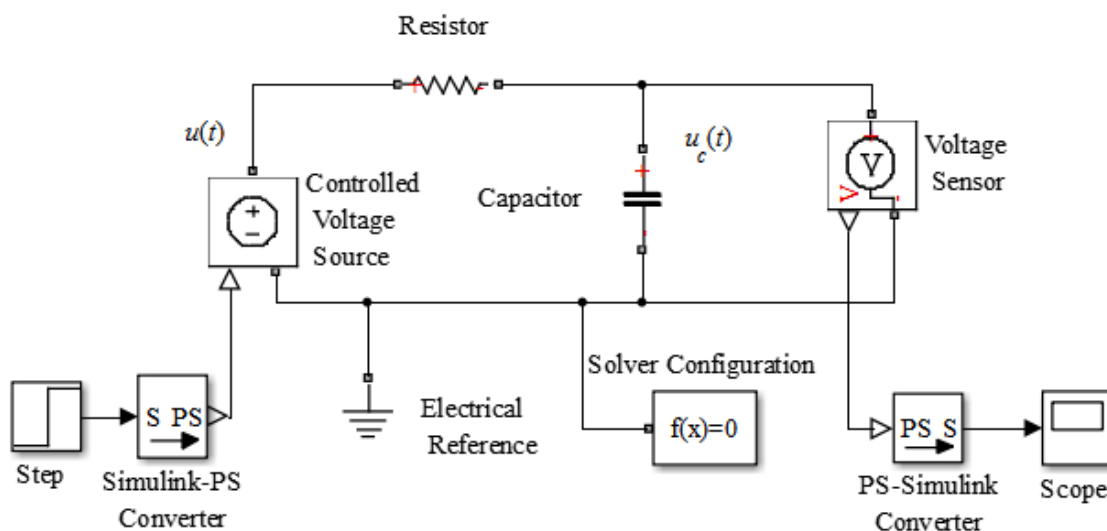
Jako ukázkový příklad je zde uveden integrační článek. V tomto příkladu si ukážeme výhody Simscape a provedeme porovnání se simulacemi s vypočítaným matematicko-fyzikálním modelem v Simulink. Integrační článek je jednoduché zapojení rezistoru R a kondenzátoru C . Kde Vstupní napětí u , je integrováno a výsledkem je napětí u_c . Pro simulace byly zvoleny následující parametry $C=32\text{nF}$, $R=100\text{k}\Omega$.



Obr. 10. Zapojení integračního článku

2.2.1 Simscape model integračního článku

Se znalostí zapojení obvodu lze vytvořit blokové zapojení v Simscape. Ze schématu je zřejmé, že budeme potřebovat jeden zdroj napětí, použijeme *Controlled Voltage Source* a jeden *Current Sensor*. Zdrojem signálu pro *Controlled Voltage Source* půjde z bloku *Step* po převedení blokem *Simulink-PS Converter*. Součástmi v obvodu budou *Resistor* a *Capacitor*. Nesmíme zapomenout na povinné součásti v obvodu to je *Solver* a *Electrical Reference*. Blok *Solver* můžeme připojit kdekoliv v obvodu.



Obr. 11. Integrační článek – Simscape knihovna

Okamžitě je tedy zřejmá výhoda Simscape knihovny bez jakékoliv námahy nebo výpočtů, kdy jsme vytvořili matematický model, který bude svým chováním odpovídat reálnému systému.

2.2.2 Diferenciální rovnice integračního článku

Nyní si integrační článek vytvoříme pomocí matematického modelu z diferenciální rovnice, při použití základních bloků Simulink. Protože pro sériové napětí odporu a kondenzátoru musí platit

$$u(t) = u_c(t) + u_R(t) \quad (1)$$

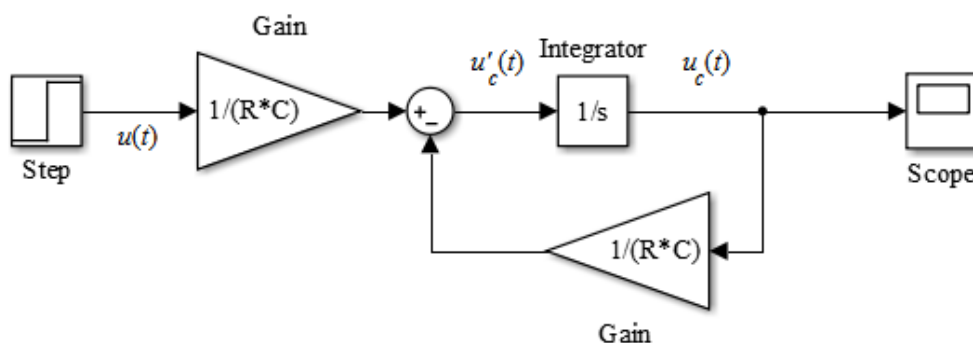
Potom po dosazení Ohmova zákona na odporu $u_R(t) = Ri_c(t)$ a dosazením proudu na kondenzátoru jako $i_c(t) = C \frac{du_c(t)}{dt}$ získáme diferenciální rovnici ve tvaru

$$RC \frac{du_c(t)}{dt} + u_c(t) = u(t) \quad (2)$$

Jak již bylo zmíněno v kapitole 1.4.1 je vhodné převést nejvyšší derivaci na jednu stranu rovnice a zbytek na druhou. Rovnice pro Simulink je tedy

$$\frac{du_c(t)}{dt} = \frac{u(t)}{RC} - \frac{u_c(t)}{RC} \quad (3)$$

Simulink schéma vytvořené podle této rovnice je na následujícím obrázku.



Obr. 12. Integrační článek – Simulink základní bloky

Je tedy zřejmé, že v tomto případě bylo třeba vytvořit matematický model, pro který bylo nutné mít znalosti zákonů elektřiny. Postup dosažení výsledku trval déle a byl složitější.

2.2.3 Přenos integračního článku

Při práci s matematickým modelem je často vhodné využít přenosu jako matematického modelu. Jeho výhodou je že už pohledem zkušeného matematika dokážeme zjistit, jak bude vypadat výstup systému. Dále lze například určit stabilitu systému. Přenos lze získat z diferenciální rovnice, kterou jsme získali v minulé kapitole.

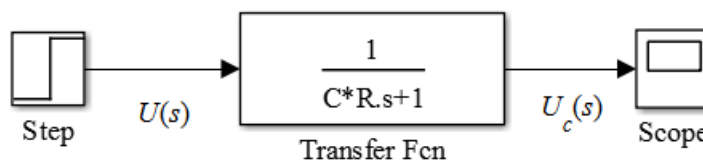
Aplikací Laplaceovy transformace na diferenciální rovnici integračního článku získáme

$$RCU_c(s)s + U_c(s) = U(s) \quad (4)$$

Z této rovnice je přenos poměrem výstupu ke vstupu, jako

$$G(s) = \frac{Y(s)}{U(s)} = \frac{U_c(s)}{U(s)} = \frac{1}{RCs + 1} \quad (5)$$

Zapojení v Simscape se provádí pomocí bloku Transfer-Fcn, výsledek je zobrazen na následujícím schématu.



Obr. 13. Integrační článek – Simulink přenos

Je zřejmé, že výpočet přenosu není až tak složitý, jako když už známe diferenciální rovnici. A je z něho možné zjistit chování systému.

2.2.4 Stavový popis integračního článku

Další možnou volbou matematického systému je stavový popis, který má podobné vlastnosti, jako přenos jen se jedná o maticový zápis.

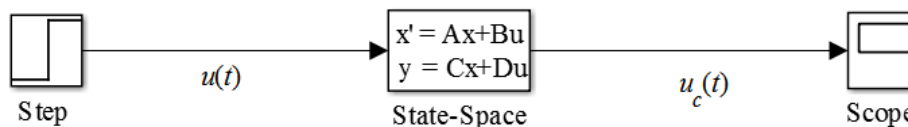
Stavový popis lze určit přímo z diferenciální rovnice jednoduchou úvahou. Je zřejmé, že rozměry všech matic budou 1x1 protože rovnice je prvního řádu. Z tohoto důvodu stačí vyjádřit první derivaci napětí na kondenzátoru z diferenciální rovnice, protože napětí na kondenzátoru bude stavovou proměnnou. Výstupní rovnice je zřejmá, protože chceme zjistit napětí na kondenzátoru, proto výstupem bude přímo stavová proměnná.

$$u'_c(t) = -\frac{1}{RC}u_c(t) + \frac{1}{RC}u(t) \quad (6)$$

$$u_c(t) = 1u_c(t)$$

Na základě získaného popisu lze provést další simulace a to přímo s využitím *State-Space*. Do bloku *State-Space* stačí zadat matice stavového popisu, to jsou v našem případě $\mathbf{A} = \left[-\frac{1}{RC}\right]$, $\mathbf{B} = \frac{1}{RC}$, $\mathbf{C} = [1]$ a $\mathbf{D} = [0]$.

Výsledné schéma v Simulink vypadá následovně

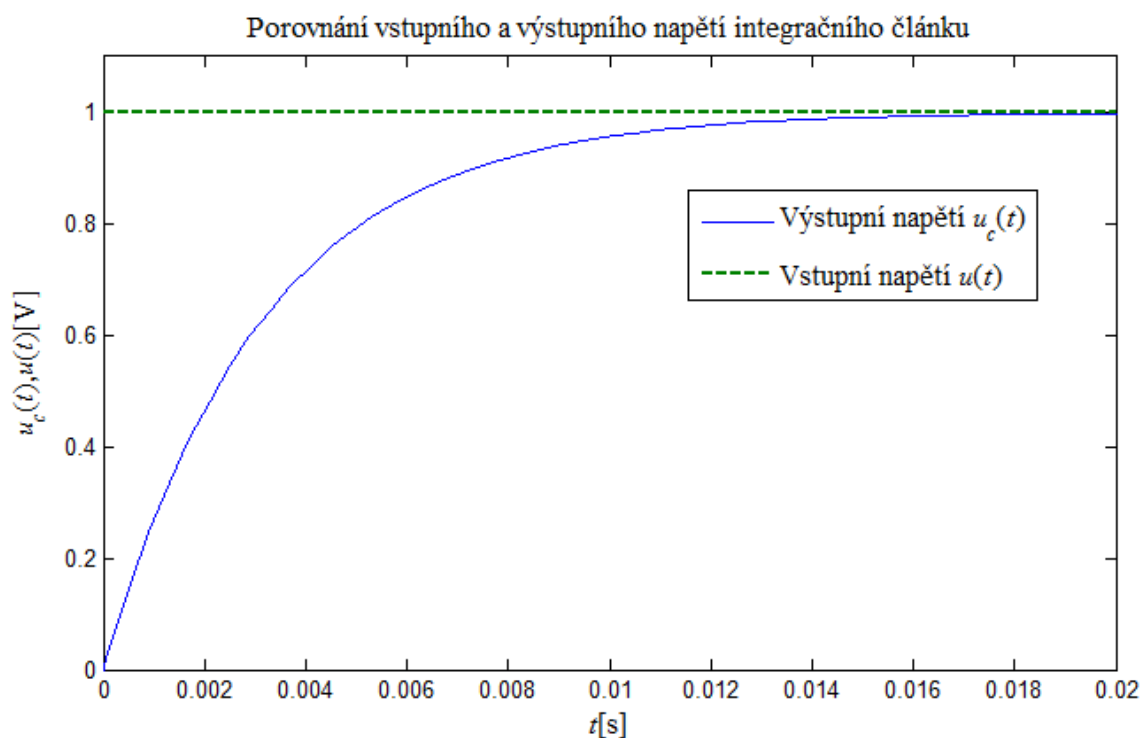


Obr. 14. Integrační článek – Stavový popis

Určení stavového popisu v tomto případě bylo jednoduché, ale nemá moc velký význam, protože jsou matice jednorozměrné, takže výsledkem je, že rovnice stavového popisu je stejná jako diferenciální rovnice.

2.2.5 Porovnání výsledků integračního článku

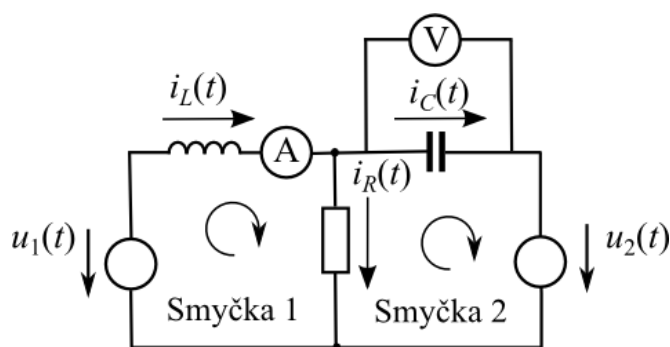
Při porovnání simulací při zadaných parametrech $C=32\text{nF}$, $R=100\text{k}\Omega$ a vstupním napětí které je nastaveno na 1V v počátku simulace, dojdeme k výsledku, že výstupy všech simulací jsou totožné. Jednoznačně nejjednodušší přístup byl v tomto případě pomocí Simscape.



Obr. 15. Výstup simulací integračního článku

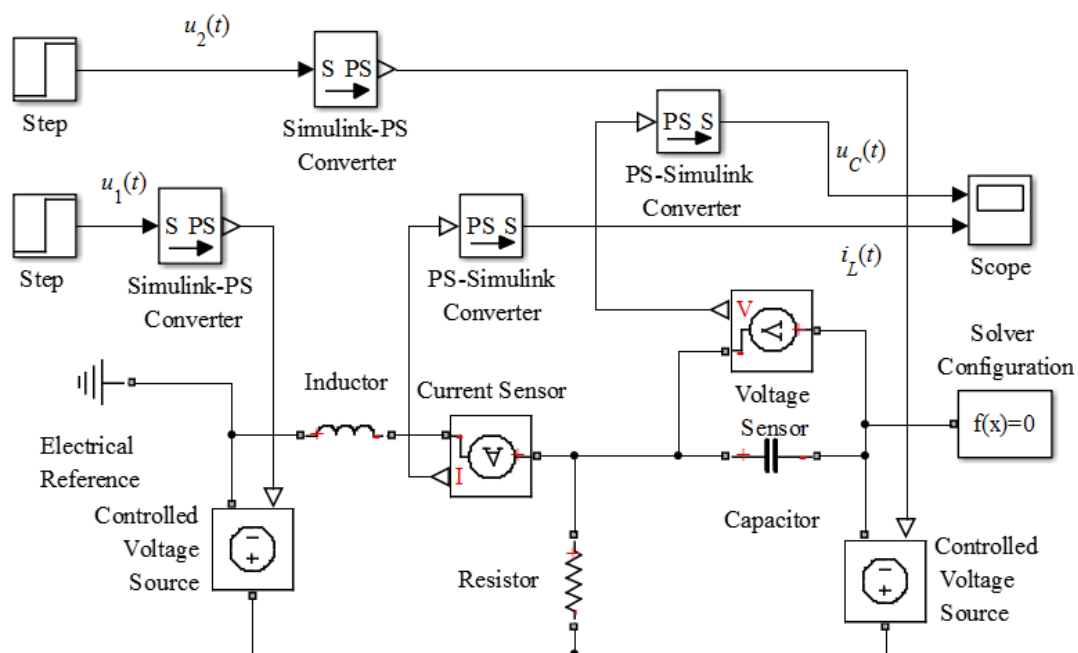
2.2.6 Ukázka řešení složitějšího elektrického obvodu v Simscape

Jedná se o elektrický obvod s dvěma vstupy $u_1(t)$ a $u_2(t)$, které jsou symbolizovány zdroji. Výstupem je proud cívkou $i_L(t)$ a napětí na kondenzátoru $u_C(t)$. Jedná se tedy o MIMO systém, protože proud cívkou a napětí na kondenzátoru jsou na sobě závislé.

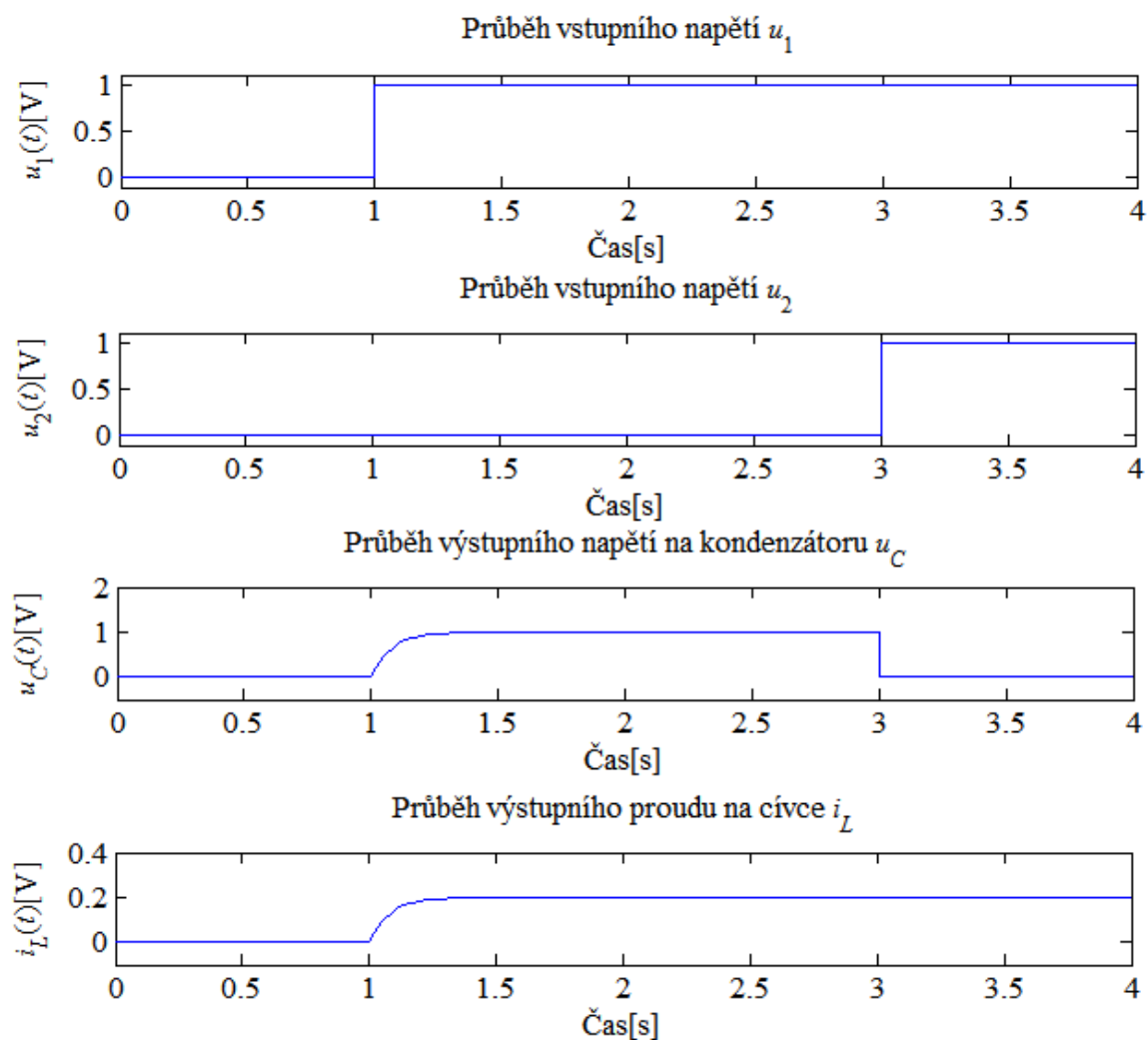


Obr. 16. Dynamický elektrický systém s dvěma zdroji

Ze znalosti zapojení můžeme přímo provést simulaci v Simscape. Jako zdroje jsou zvoleny *Controlled Voltage Source* s přívodem signálu z Simulink. K obvodu je nutné dále připojit Solver a referenční bod. Použité komponenty jsou *Capacitor*, *Inductor* a *Resistor*. Pro detekci napětí je využito *Current sensor*. Nesmí se zopomenout na *Electrical Reference* a *Solver Configuration*.



Obr. 17. Dynamický elektrický systém s dvěma zdroji

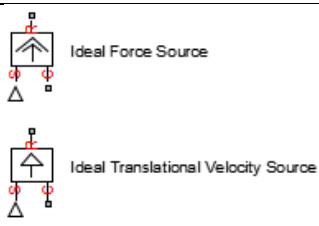
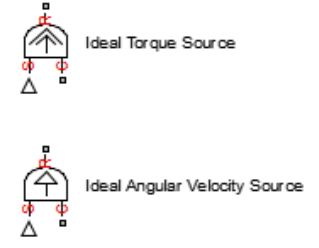
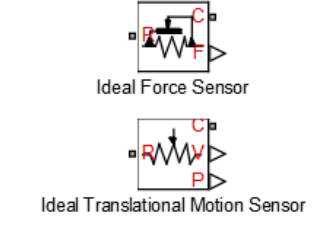


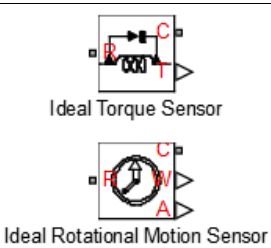
Obr. 18. Průběhy obvodu s dvěma zdroji a dynamickými prvky

Je zřejmé, že zapojení s knihovnou Simscape lze vytvořit velmi jednoduše a zároveň jsou zachovány propojení jako v reálném systému.


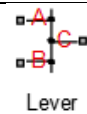

2.3 Základní knihovna mechanických součástí Simscape

Knihovna Simscape obsahuje mnoho součástí, které jsou používány v mechanických systémech. Obsahuje jak translační, tak rotační mechanické prvky. Obsahuje zdroje rychlosti síly, momentu a úhlu. Obsahuje sensory síly, polohy, momentu a úhlu. Jako hlavní součásti obvodu lze použít hmoty, pružiny, tření, tlumení, převodovky a páky, dle [5]. Většina z těchto součástí je implementována, jako ideální součásti (jinak je to u nich uvedeno).








Simscape bloky mechanických zdrojů a sensorů	
Simscape blok	Základní popis
 <p>Diagram showing two blocks: 'Ideal Force Source' and 'Ideal Translational Velocity Source'. Both blocks have a square input port on the left and a circular output port on the right. The 'Ideal Force Source' block has a red 'F' label, and the 'Ideal Translational Velocity Source' block has a red 'V' label.</p>	<p>Bloky traslačních zdrojů síly/rychlosti jsou</p> <ul style="list-style-type: none"> • <i>Ideal Force Source</i> je zdrojem síly, má jeden vstup pro hodnotu síly ze Simulink. • <i>Ideal Translational Velocity Source</i> je zdroj rychlosti, má jeden vstup pro hodnotu rychlosti ze Simulink.
 <p>Diagram showing two blocks: 'Ideal Torque Source' and 'Ideal Angular Velocity Source'. Both blocks have a square input port on the left and a circular output port on the right. The 'Ideal Torque Source' block has a red 'T' label, and the 'Ideal Angular Velocity Source' block has a red 'W' label.</p>	<p>Bloky rotačních zdrojů momentu/úhlové rychlosti jsou</p> <ul style="list-style-type: none"> • <i>Ideal Torque Source</i> je zdroj momentu, má jeden vstup pro hodnotu momentu ze Simulink. • <i>Ideal Angular Velocity Source</i> je zdroj úhlové rychlosti, má jeden vstup pro hodnotu úhlové rychlosti ze Simulink.
 <p>Diagram showing two blocks: 'Ideal Force Sensor' and 'Ideal Translational Motion Sensor'. Both blocks have a square input port on the left and two circular output ports on the right. The 'Ideal Force Sensor' block has a red 'F' label, and the 'Ideal Translational Motion Sensor' block has a red 'V' and 'P' label.</p>	<p>Bloky translačních sensorů síly/rychlosti/pozice jsou</p> <ul style="list-style-type: none"> • <i>Ideal Force Sensor</i> je snímač síly, který má výstup s hodnotou síly do Simulink • <i>Ideal Translational Sensor</i> je snímač polohy/rychlosti, který má dva výstupy do Simulink výstup V je rychlosti a výstup P je pozice.

 <p>Diagram showing two Simulink blocks: 'Ideal Torque Sensor' and 'Ideal Rotational Motion Sensor'. The torque sensor has inputs F and C, and output W. The rotational motion sensor has inputs F and C, and outputs W and A.</p>	<p>Bloky rotačních sensorů momentu/úhlové rychlosti/úhlu jsou</p> <ul style="list-style-type: none"> • <i>Ideal Torque Sensor</i> je snímač momentu, který má výstup s hodnotou síly do Simulink • <i>Ideal Rotational Motion Sensor</i> je snímač úhlu/úhlové rychlosti, který má dva výstupy do Simulink výstup W je úhlová rychlosti a výstup A je úhel.
---	---

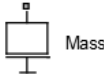
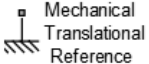
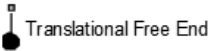




Tab. 13. Simscape bloky mechanických zdrojů a sensorů

Simscape bloky mechanismů	
Simscape blok	Základní popis
 <p>Gear Box</p>	<p>Blok <i>Gear box</i> je blok symbolizující převodovku, má jeden zadavatelný parametr a to parametr převodu momentu kterým může být kladné nebo záporné číslo.</p>
 <p>Lever</p>	<p>Blok <i>Lever</i> představuje páku, jedná se vlastně o tyč, na které jsou 3 body A, B a C. V případě že bod C připojíme na zem, potom získáme dvojzvratnou páku mezi bodem A a B. V případě připojení země k bodu A nebo B získáme jednozvratnou páku. Parametry bloku jsou vzdálenost mezi A a C, a dalším parametrem je vzdálenost mezi B a C.</p>
 <p>Wheel and Axle</p>	<p>Blok <i>Wheel and Axle</i> slouží jako převodník mechanického rotačního pohybu na translační nebo naopak. V bloku je možné nastavení průměru kola a směr, kterým bude translační pohyb.</p>

Tab. 14. Simscape bloky mechanismů

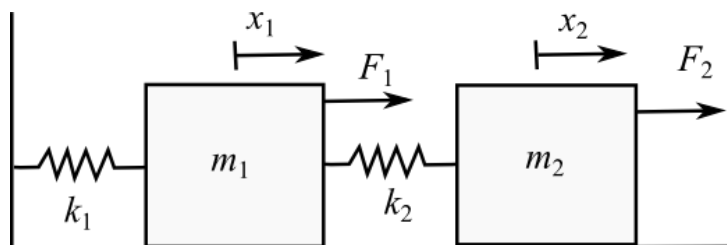
Simscape bloky mechanických rotačních součástí	
Simscape blok	Základní popis
 Inertia	Blok <i>Inertia</i> symbolizuje rotační hmotu s nastavitelným momentem setrvačnosti a počáteční rychlostí.
 Mechanical Rotational Reference	Blok <i>Mechanical Rotational Reference</i> slouží jako referenční bod mechanických systémů.
 Rotational Free End	Blok <i>Rotational Free End</i> slouží jako volný konec mechanického obvodu (neuzemněný).
 Rotational Friction	Blok <i>Rotational Friction</i> slouží pro přidání rotačního tření mezi dvěma hřídelemi. Tento blok umožňuje nastavení Coulubuvského tření, tření při roztáčení.
 Rotational Spring	Blok <i>Rotational Spring</i> je blokem torzní pružiny a je možné nastavit její tuhost a počáteční deformaci.
 Rotational Damper	Blok <i>Rotational Damper</i> je rotační tlumič u kterého je možné nastavit jeho koeficient útlumu.
 Rotational Hard Stop	Blok <i>Rotational Hard Stop</i> slouží jako omezení rotačního pohybu. Kdy pohyb hřídele R je omezen zábranou C, která je implementována jako pružná zábrana. Je možné nastavit pozice zábran na obou stranách, počáteční pozici. Dále je možné nastavit útlum způsobený zábranou a její tuhost.

Tab. 15. Simscape bloky rotačních mechanických součástí

Simscape bloky mechanických translačních součástí	
Simscape blok	Základní popis
 Mass	Blok <i>Mass</i> symbolizuje hmotu, která má nastavitelnou hmotnost a počáteční rychlost.
 Mechanical Translational Reference	Blok <i>Mechanical Translational Reference</i> slouží jako referenční bod rotačních mechanických systémů.
 Translational Free End	Blok <i>Translational Free End</i> slouží jako volný konec (neuzeměný).
 Translational Friction	Blok <i>Translational Friction</i> slouží pro přidání tření mezi dvě plochy. Tento blok umožňuje nastavení Coulubuvského tření, tření při počátku pohybu.
 Translational Spring	Blok <i>Translational Spring</i> slouží jako translační pružina s nastavitelnou tuhostí a počáteční deformací.
 Translational Damper	Blok <i>Translational Damper</i> slouží jako tlumič s nastavitelným koeficientem útlumu.
 Translational Hard Stop	Blok <i>Translational Hard Stop</i> slouží jako omezení translačního pohybu mezi dvěma koncovými body, které jsou implementované jako pružné. Je možné nastavit pozice koncových bodů, počáteční stav. Jejich tuhost a jejich útlum.

Tab. 16. Simscape bloky mechanických translačních součástí

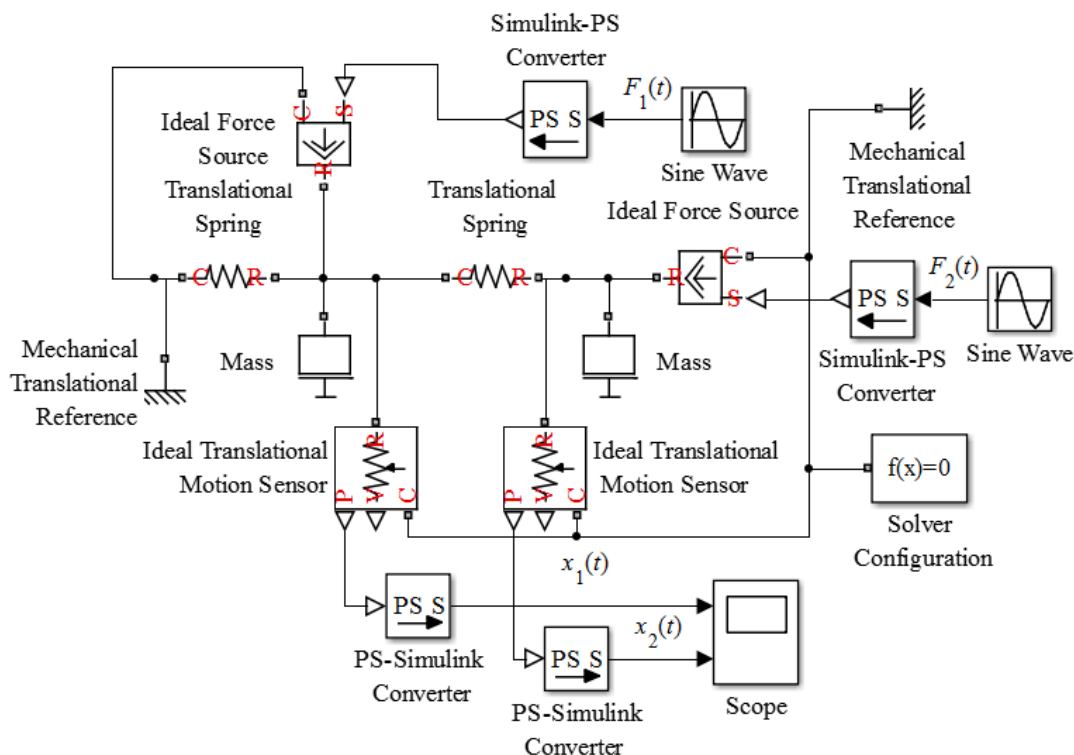
Jako příklad mechanických systémů zde uvedeme systém dvou hmot m_1 a m_2 , které jsou propojené pružinami o koeficientu tuhosti b_1, b_2 z nichž jedna je pevně zakotvena. Tyto hmoty jsou na vodorovné podložce bez tření. Na tento systém působí vnější síla F_1 na hmotu m_1 a F_2 na hmotu m_2 . Pozice hmoty m_1 je x_1 a pozice hmoty m_2 je x_2 .



Obr. 19. Hmoty propojené pružinou

2.3.1 Simscape model dvou hmot propojených pružinou

Se znalostí propojení součástí ze schématu, lze přejít k simulaci s knihovnou Simscape. Využije se základních mechanických prvků obsažených v tomto systému, jako je pružina odpovídající bloku *Translational Spring* a hmota odpovídající bloku *Mass*. Zdrojem síly bude *Ideal Force Source* na který bude přiveden sinusový signál ze Simulink přes konvertor. Jako sensory jsou využity *Ideal Translational Motion Sensor* bloky. Výsledné Simscape schéma je na následujícím obrázku.



Obr. 20. Hmoty propojené pružinou – Simscape

Je zřejmé, že zapojení je přesně podle schématu a je velice jednoduché.

2.3.2 Diferenciální rovnice dvou hmot propojených pružinou

Při tvorbě diferenciálních rovnic je možné vyjít druhého Newtonova zákona $ma = \sum F$.

Pro tento systém bude tedy platit:

$$m_1 \ddot{x}_1 = -k_1 x_1 + k_2(x_2 - x_1) + F_1 \quad (7)$$

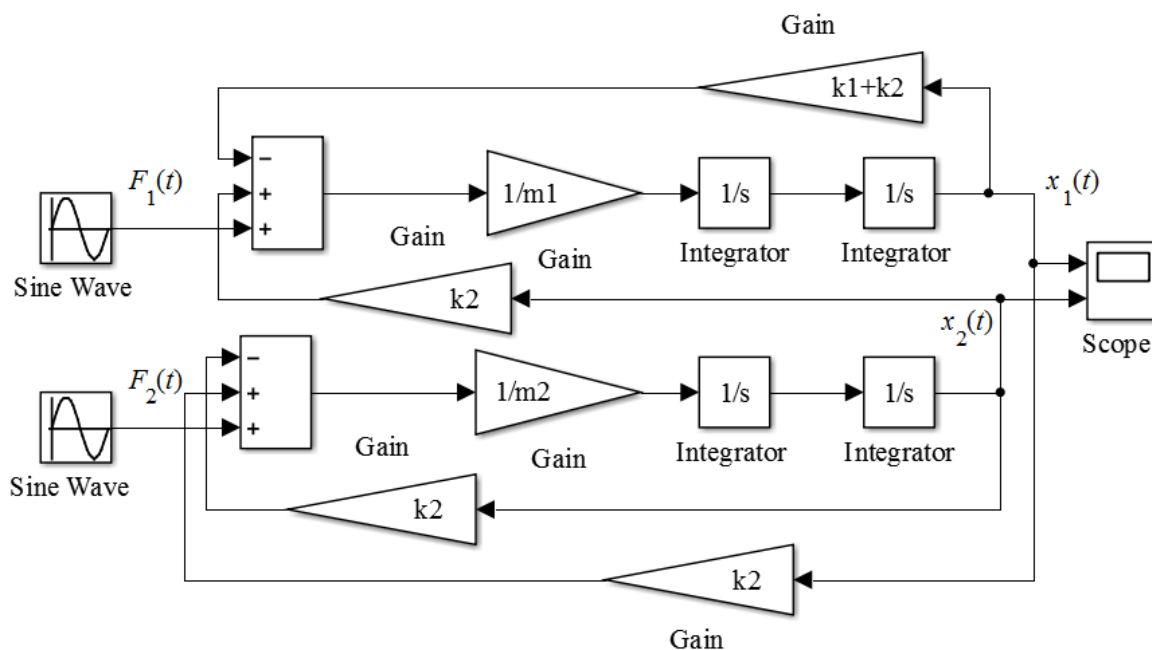
$$m_2 \ddot{x}_2 = k_2(x_1 - x_2) + F_2 \quad (8)$$

Tyto diferenciální rovnice převedeme do tvaru pro Simulink, jako

$$\ddot{x}_1 = -\frac{k_1 x_1}{m_1} + \frac{k_2(x_2 - x_1)}{m_1} + \frac{F_1}{m_1} \quad (9)$$

$$\ddot{x}_2 = \frac{k_2(x_1 - x_2)}{m_2} + \frac{F_2}{m_2} \quad (10)$$

Výsledné Simulink schéma je potom zobrazeno na následujícím obrázku



Obr. 21. Hmoty propojené pružinou – Diferenciální rovnice

Získání diferenciálních rovnic je v tomto případě relativně jednoduché ale lze v nich udělat chybu a výsledné ze Simulink schématu nemusí být jednoznačně jasný jeho význam.

2.3.3 Stavový popis dvou hmot propojených pružinou

Stavový popis lze odvodit z diferenciálních rovnic úvahou na základě, že derivací dráhy podle času je rychlost, derivací rychlosti podle času je zrychlení. Protože, je tento systém tvořen dvěma rovnicemi druhého řádu, musí se zavést 4 stavové proměnné. Necht' jsou tedy zavedeny stavové proměnné x_1 a x_2 , jejichž první derivace bude odpovídat zrychlení \ddot{x}_1 a \ddot{x}_2 . Dále jsou zavedeny stavové proměnné x_3 a x_4 , jejichž první derivace bude odpovídat rychlosti \dot{x}_1 a \dot{x}_2 . Platí tedy substituce

$$x_1 = x_1, x_2 = x_2, x_3 = \dot{x}_1, x_4 = \dot{x}_2 \quad (11)$$

Tím že byly zavedeny stavové proměnné tímto způsobem, budou určitě mít všechny stavy systému fyzikální význam. Diferenciální rovnice lze upravit do lepšího tvaru, aby přepis na stavový popis byl zřetelnější

$$\ddot{x}_1 = -\frac{x_1(k_1 + k_2)}{m_1} + \frac{k_2 x_2}{m_1} + \frac{F_1}{m_1} \quad (12)$$

$$\ddot{x}_2 = \frac{k_2 x_1}{m_2} - \frac{k_2 x_2}{m_2} + \frac{F_2}{m_2} \quad (13)$$

Výsledný stavový popis z těchto rovnic potom bude

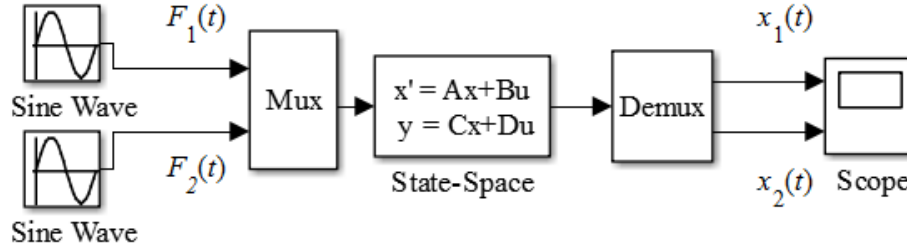
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1 + k_2}{m_1} & \frac{k_2}{m_1} & 0 & 0 \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \quad (14)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (15)$$

Jednotlivé matice stavového popisu jsou tedy

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1 + k_2}{m_1} & \frac{k_2}{m_1} & 0 & 0 \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \text{ a } D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Se znalostí stavového popisu lze vytvořit zapojení v MATLAB/Simulink s využitím bloku *State-Space*. Dále s využitím bloku *Mux* pro vektorizaci vstupu a *Demux* pro rozdělení vektoru výstupu na jednotlivé prvky.



Obr. 22. Hmoty propojené pružinou – Stavový popis

Odvození stavového popisu je v tomto případě jednoduché protože vychází přímo z diferenciálních rovnic.

2.3.4 Přenos dvou hmot propojených pružinou

Přenos lze získat jako Laplaceovu transformaci rovnice stavového popisu. Po dosazení za jednotlivé matice získáme

$$\mathbf{G}(s) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} s - \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_1 + k_2}{m_1} & \frac{k_2}{m_1} & 0 & 0 \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & 0 & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix}$$

Po provedení matematických úprav získáme přenosovou matici ve tvaru

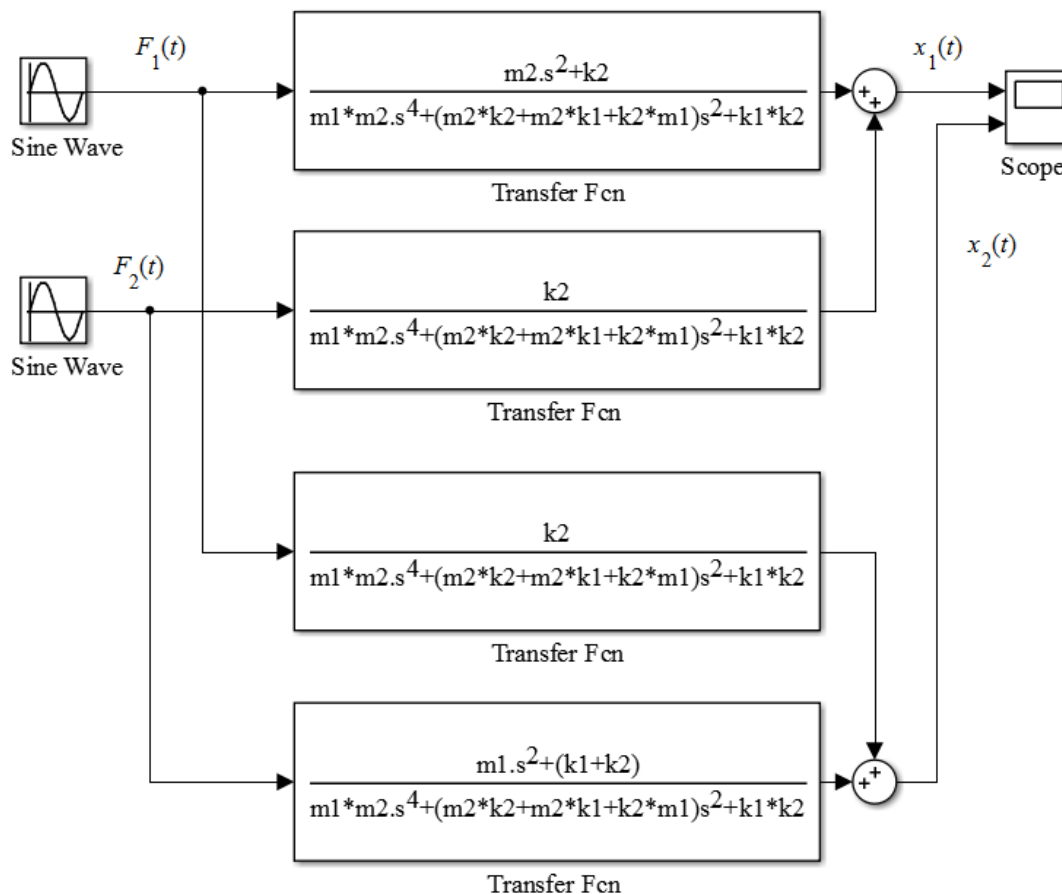
$$\mathbf{G}(s) = \begin{bmatrix} \frac{m_2 s^2 + k_2}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2) s^2 + k_2 k_1} & \frac{k_2}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2) s^2 + k_2 k_1} \\ \frac{k_2}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2) s^2 + k_2 k_1} & \frac{m_1 s^2 + k_2 + k_1}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2) s^2 + k_2 k_1} \end{bmatrix}$$

Rozdělením této matice na dílčí přenosy lze provést další simulaci v MATLAB/Simulink. Dílčí přenosy jsou získány rozepsáním maticového přenosu na dvě rovnice, které se budou řešit v MATLAB/Simulink

$$\begin{aligned} X_1(s) = & \frac{m_2 s^2 + k_2}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2)} F_1(s) \\ & + \frac{k_2}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2)} F_2(s) \end{aligned} \quad (16)$$

$$X_2(s) = \frac{k_2}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2)} F_1(s) + \frac{m_1 s^2 + k_2 + k_1}{m_1 m_2 s^4 + (k_2 m_1 + k_2 m_2 + k_1 m_2)} F_2(s) \quad (17)$$

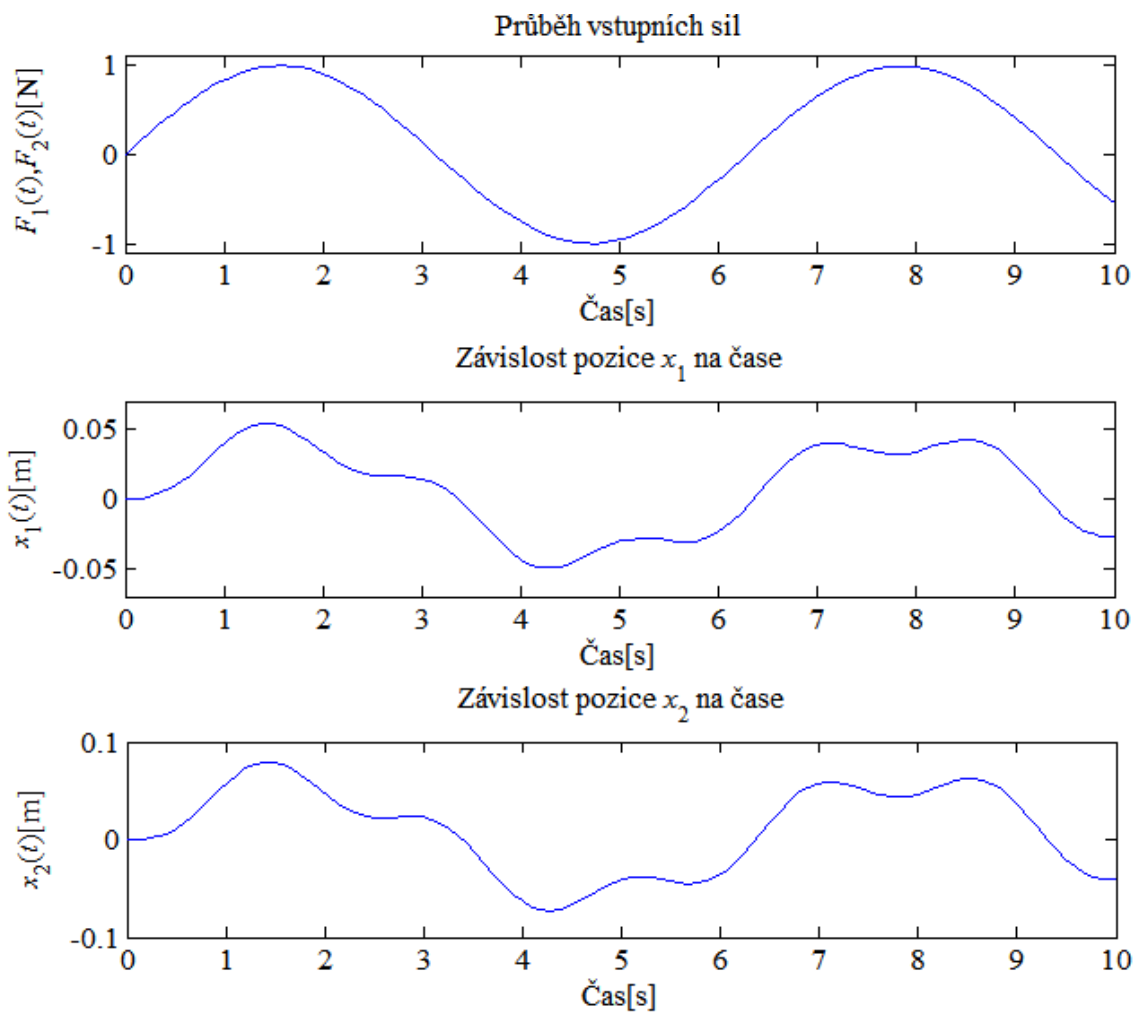
V této simulaci se využije bloků *Transfer Fcn* a *Sum*. Zdrojem bude *Sine Wave*, a data se zobrazí pomocí bloku *Scope*.



Obr. 23. Hmoty propojené pružinou – Přenos

2.3.5 Porovnání výsledků simulací dvou hmot propojených pružinou

Při porovnání simulací při zadaných parametrech $k_1=50\text{N/m}$, $k_2=60\text{N/m}$, $m_1=1\text{kg}$, $m_2=2\text{kg}$. Vstupní síla F_1 a F_2 bude odpovídat sinusovému průběhu s amplitudou 1N a frekvencí 1Hz jsme dosáhly stejných výsledků u všech matematických modelů.

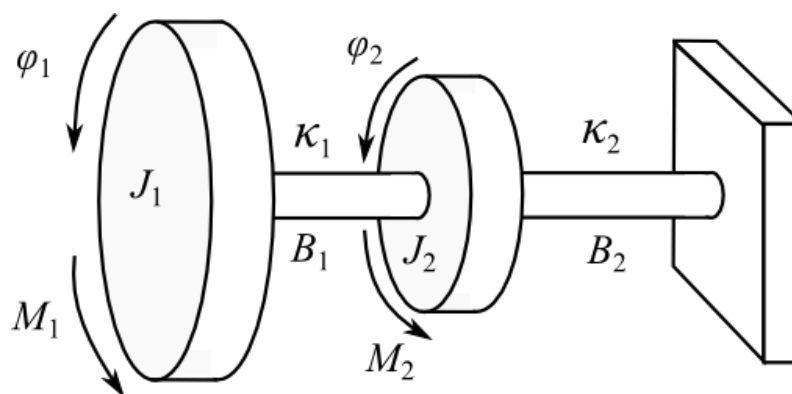


Obr. 24. Porovnání simulací dvou hmot s pružinou

Jednoznačně nejjednodušší přístup byl opět pomocí Simscape.

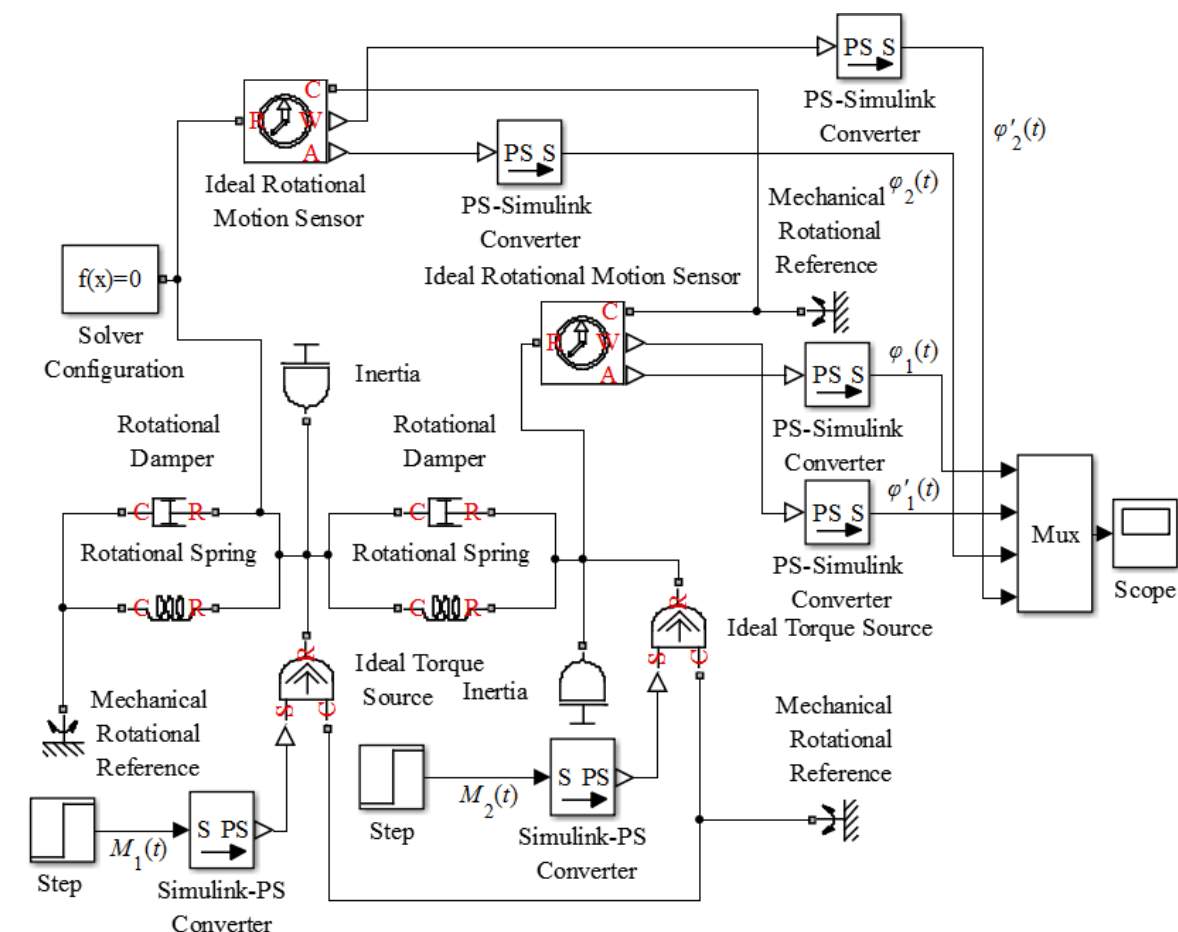
2.3.6 Složitější mechanický příklad

Jde o systém se dvěma vstupy a výstupy. Systém je tvořen dvěma rovnoměrně rozloženými hmotami s momentem setrvačnosti J_1 a J_2 . Tyto hmoty jsou propojeny trubkou, která obsahuje torzní pružinu a tlumič, stejně propojená je i druhá hmota se stěnou, která je pevně ukotvená. Koeficient tuhosti pružiny mezi hmotami je κ_1 a tlumič mezi hmotami má koeficient tlumení B_1 . Koeficient tuhosti pružiny mezi druhou hmotou a stěnou je označen κ_2 a tlumič mezi druhou a hmotou má koeficient tlumení B_2 . Vstupy do systému jsou zajištěny pomocí momentů působících na hmoty, moment M_1 na první hmotu a moment M_2 na druhou. Výstupem z tohoto systému jsou úhly natočení hmot jako φ_1 a φ_2 .



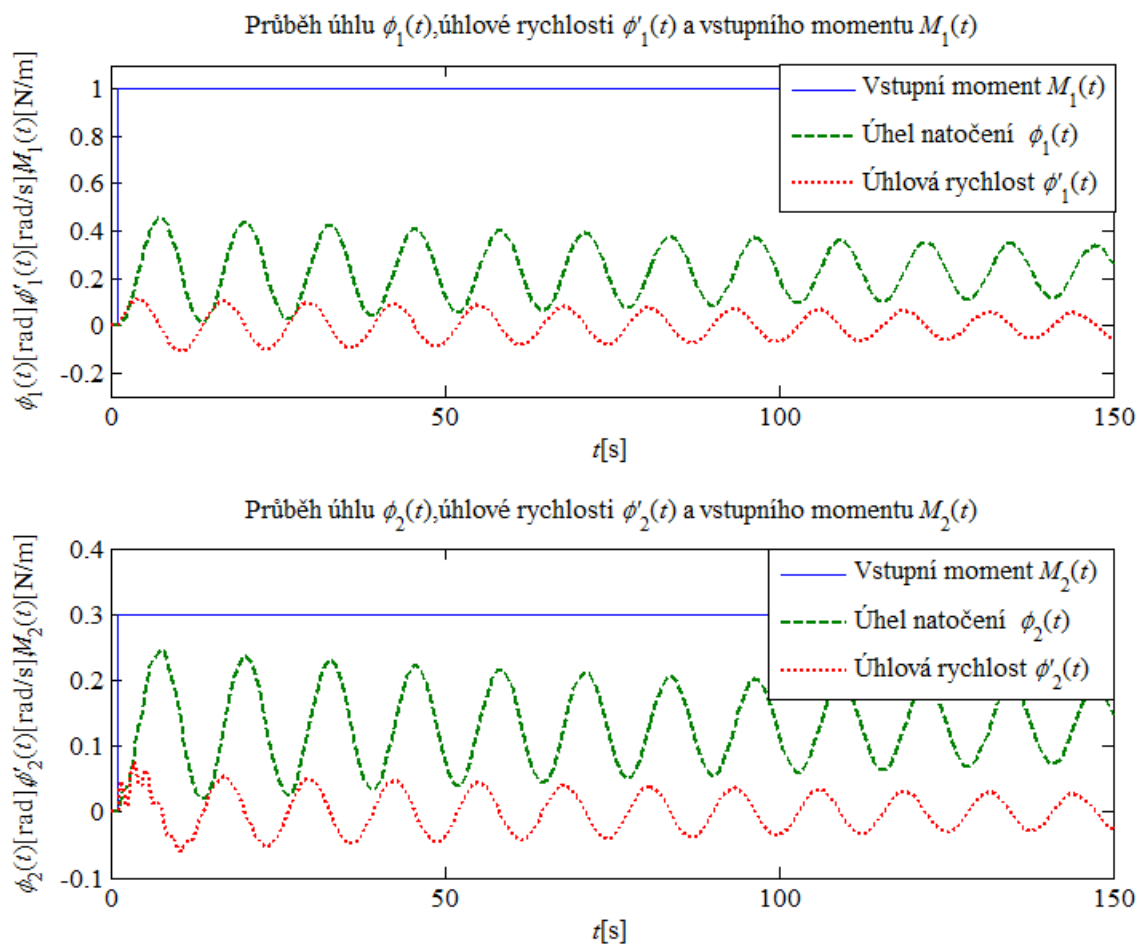
Obr. 25. Rotační mechanický systém

Se znalostí struktury mechanického modelu lze provést zapojení s použitím knihovny Simscape. Pro zapojení se dá využít bloků *Inertia*, *Rotational Spring*, *Rotational Damper* a *Mechanical Rotational Reference*. Vstup do systému je proveden pomocí bloků *Step* a převodu na Simscape signál který je přiveden na *Ideal Torque Source*. Detekce úhlu je zajištěna pomocí bloků *Ideal Rotational Motion Sensor*, kde po převedení signálu se výsledky zobrazí na bloku *Scope*.



Obr. 26. Rotační mechanický systém – Simscape

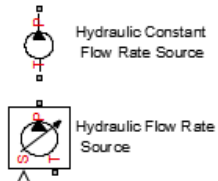
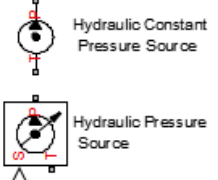
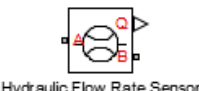
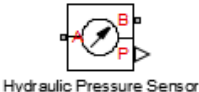
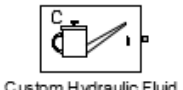
Opět je zřejmé že zapojení je přímo podle schématu a dá se v něm dobře vyznat.









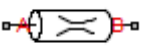


Obr. 27. Průběhy veličin rotačního systému


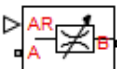

2.4 Základní knihovna hydraulických součástí Simscape

Knihovna Simscape obsahuje mnoho částí, které jsou používány v hydraulických systémech. Obsahuje jak translační, tak rotační mechanické prvky. Obsahuje zdroje rychlosti síly, momentu a úhlu. Obsahuje ventily, odporové trubky spády kapaliny, tlakové nádrže, rotační a translační hydromechanické motory a mnoho dalšího, dle [5]. Většina z těchto součástí je implementována, jako ideální součásti (jinak je to u nich uvedeno).

Simscape bloky hydraulických zdrojů a sensorů	
Simscape blok	Základní popis
 <p>Hydraulic Constant Flow Rate Source</p> <p>Hydraulic Flow Rate Source</p>	<p>Mezi zdroje průtoku patří bloky</p> <ul style="list-style-type: none"> • <i>Hydraulic Constant Flow Rate Source</i> je blok, který poskytuje konstantní objemový průtok z nastaveného průtoku. • <i>Hydraulic Flow Rate Source</i> je blok, který poskytuje objemový průtok podle připojeného signálu ze Simulink.
 <p>Hydraulic Constant Pressure Source</p> <p>Hydraulic Pressure Source</p>	<p>Mezi zdroje tlaku patří bloky</p> <ul style="list-style-type: none"> • <i>Hydraulic Constant Pressure Source</i> je blok, který poskytuje konstantní tlak podle nastaveného parametru. • <i>Hydraulic Pressure Source</i> je blok, který poskytuje tlak v závislosti na připojeném signálu ze Simulink.
 <p>Hydraulic Flow Rate Sensor</p>	<p>Blok <i>Hydraulic Flow Rate Sensor</i> slouží pro měření objemového průtoku.</p>
 <p>Hydraulic Pressure Sensor</p>	<p>Blok <i>Hydraulic Pressure Sensor</i> slouží pro měření tlaku.</p>
 <p>Custom Hydraulic Fluid</p>	<p>Blok <i>Custom Hydraulic Fluid</i> umožňuje definovat vlastnosti vlastní kapaliny v obvodu. Parametry jsou hustota, kinematická viskozita a další.</p>

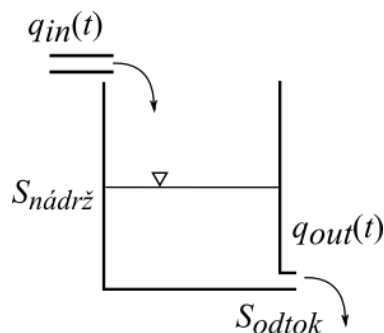
Tab. 17. Simscape bloky hydraulických zdrojů a sensorů

Simscape bloky hydraulických součástí	
Simscape blok	Základní popis
 Constant Area Hydraulic Orifice	<p>Blok <i>Constant Area Hydraulic Orifice</i> je součást, která dočasně zužuje průtočný průměr, pro změnu průtoku. Parametrem je zúžený průřez.</p>
 Constant Volume Hydraulic Chamber	<p>Blok <i>Constant Volume Hydraulic Chamber</i> je součást která umožňuje uchování hydraulické energie. Parametry jsou obsah a počáteční tlak.</p>
 Fluid Inertia	<p>Blok <i>Fluid Inertia</i> reprezentuje změnu tlaku způsobenou změnami rychlostí při průtoku trubkou.</p>
 Hydraulic Cap	<p>Blok <i>Hydraulic Cap</i> zamezuje průtoku média.</p>
 Hydraulic Piston Chamber	<p>Blok <i>Hydraulic Piston Chamber</i> je komora s pístem do které se ukládá hydraulická energie. Definuje se velikost nádrže plocha pístu a počáteční tlak.</p>
 Hydraulic Reference	<p>Blok <i>Hydraulic Reference</i> je referenční body hydraulických systémů.</p>
 Hydraulic Resistive Tube	<p>Blok <i>Hydraulic Resistive Tube</i> je blok který symbolizuje odporové ztráty vedením kapaliny trubkou. Nastavuje se tvar trubky, délka, hrubost vnitřního povrchu.</p>
 Linear Hydraulic Resistance	<p>Blok <i>Linear Hydraulic Resistance</i> představuje hydraulický odpor kde vzniklý objemový průtok je přímo úměrný ztrátě tlaku.</p>
 Rotational Hydro-Mechanical Converter	<p>Blok <i>Rotational Hydro-Mechanical Converter</i> slouží jako převodník hydraulické energie na rotační mechanickou energii, nebo naopak. Jako součást Simscape je to jedno lamelový kývný hydromotor. Kde port A je vstup pro hydraulickou kapalinu a port C, S jsou porty pro rotačně mechanické.</p>

 Translational Hydro-Mechanical Converter	<p>Blok <i>Translational Hydro-Mechanical Converter</i> slouží jako převodník hydraulické energie na translační mechanickou energii, nebo naopak. Jako součást Simscape je to jednočinný hydromotor. Kde A je port pro hydraulickou kapalinu a porty A a R jsou mechanické porty.</p>
 Variable Area Hydraulic Orifice	<p>Blok <i>Variable Area Hydraulic Orifice</i> je blok, kde rychlost průtoku je přímo úměrná rozdílu tlaků vzniklým rozdílem průřezů, kde tento průřez je ovladatelný signálem ze Simulink.</p>
 Variable Hydraulic Chamber	<p>Blok <i>Variable Hydraulic Chamber</i> je blok, který reprezentuje nádrž s proměnným objemem a pístem ovládaným mechanicky pomocí signálu ze Simulink na portu V.</p>

Tab. 18. Simscape bloky hydraulických součástí

Jako příklad hydraulického systému je jednoduchá nádrž s jedním vstupem, přítokem $q_{in}(t)$ a jedním výstupem, v podobě hladiny $h(t)$. Nádrž je popsána plochou průřezu $S_{nádrž}$ a plochou průřezu výtoku S_{odtok} . Výtok z nádrže je označen $q_{out}(t)$.



Obr. 28. Nádrž

V tomto případě dojdeme na to, že neexistuje blok, který by byl ekvivalentní tomuto příkladu. Z tohoto důvodu v současné době nedokážeme vytvořit Simscape schéma. Proto, abychom mohly vytvořit Simscape schéma, musíme nejdřív vytvořit vlastní blok nádrže, pro který potřebujeme znát diferenciální rovnice.

2.4.1 Diferenciální rovnice nádrže

Za pomoci bilance lze získat rovnici popisující nádrž a to jako

$$q_{in} = q_{out} + S_{nádrž} \frac{dh(t)}{dt} \quad (18)$$

Pro nádrž a otvor musí platit zákon o zachování energie v podobě rovnice kontinuity.

$$S_{nádrž}v = S_{odtok}v_{odtok} \quad (19)$$

Z rovnice kontinuity se dá odvodit výtoková rychlost a to pomocí Bernulliho. Bernulliho rovnice popisuje zákon zachování mechanické energie, kde bude platit, že součet kinetické, potenciální a gravitační energie je roven konstantě

Z toho že platí zákon zachování mechanické energie lze usoudit, že energie před výtokem z nádrže a po výtoku bude stejná. Pro nádrž potom platí

$$\frac{v}{2} + \frac{p}{\rho} + gh = \frac{v_{odtok}}{2} + \frac{p_{odtok}}{\rho} + gh_z \quad (20)$$

kde hodnoty v , p jsou hodnoty před odtokem, hodnoty v_{odtok} , p_{odtok} po odtoku a h_z představuje ztrátu vzniklou ve výtokovém otvoru. Po dosazení rovnice kontinuity do Bernulliho rovnice a dosazení za $h_z = \xi \frac{v_{odtok}^2}{2g}$ a vyjádřením výtokové rychlosti, kterou je nutná pro vytvoření matematického modelu, získáme jako

$$v_{odtok} = \sqrt{2 \frac{\frac{p - p_{odtok}}{\rho} + gh}{1 + \xi - \left(\frac{S_{odtok}}{S_{nádrž}}\right)^2}} \quad (21)$$

Protože se počítá s dokonalou kapalinou, ztrátový součinitel bude tedy $\xi = 0$. Dále je zřejmé, že nádrž má zásadně větší průřez než odtokový otvor proto můžeme úplně vyloučit člen $\left(\frac{S_{odtok}}{S_{nádrž}}\right)^2$. Všechny nádrže jsou otevřeny do atmosféry, proto bude přibližně platit $p \cong p_{odtok}$, dle [3]. Proto jako výslednou rovnici pro rychlost dostaneme

$$v_{odtok} = \sqrt{2gh} \quad (22)$$

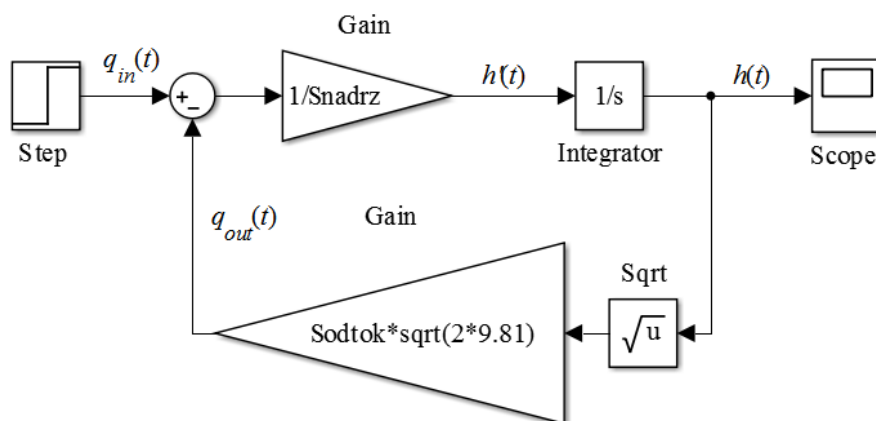
Se znalostí výtokové rychlosti lze určit objemový průtok kapaliny, který vyteče z nádrže. Po dosazení do bilanční rovnice vznikne matematický model nádrže, jako

$$q_{in} = S_{odtok} \sqrt{2gh} + S_{nádrž} \frac{dh(t)}{dt} \quad (23)$$

Vyjádřením nejvyšší derivace vznikne rovnice vhodná pro použití ve zpětnovazebním obvodu MATLAB/Simulink.

$$\frac{dh(t)}{dt} = \frac{q_{in}}{S_{nádrž}} - \frac{S_{odtok}}{S_{nádrž}} \sqrt{2g} * \sqrt{h} \quad (24)$$

Výsledný simulační obvod ze základních součástí v Simulink je zobrazen na následujícím obrázku.



Obr. 29. Nádrž – Diferenciální rovnice

2.4.2 Simscape model nádrže

Vytvoříme vlastní Simscape blok s použitím „*Simscape language*“. Vysvětlení postupu je v kapitole 3. Pro vytvoření nového modelu je nutné vytvořit novou knihovnu bloků, a to lze provést vytvořením složky kdekoliv na disku, kde chceme mít uloženou knihovnu. Název složky musí začínat znakem „+“. Všechny zdrojové kódy obsažené v této složce budou součástí knihovny s názvem složky. Nový zdrojový kód vytvoříme v menu MATLAB výběrem „*New>Script*“. Nově vytvořený soubor okamžitě uložíme do vytvořené složky budoucí knihovny. Soubor uložíme pod názvem bloku, který vytváříme a s koncovkou „.ssc“. Po uložení s touto koncovkou se bude zvýrazňovat syntax „*Simscape language*“. Každý zdrojový kód bloku Simscape začíná názvem bloku, kde může být uvedena fyzikální doména všech portů, pokud je jednotná, dále se uvedou parametry bloku, vstupy a výstupy pro MATLAB/Simulink. Proměnné a jejich vliv na vstupy a výstupy bloku a nakonec diferenciální rovnice. Pro matematický model si vyjádříme hladinu jednou rovnicí tak abychom mohli zjistit její okamžitou hodnotu, druhou rovnicí určíme odtok z nádrže.

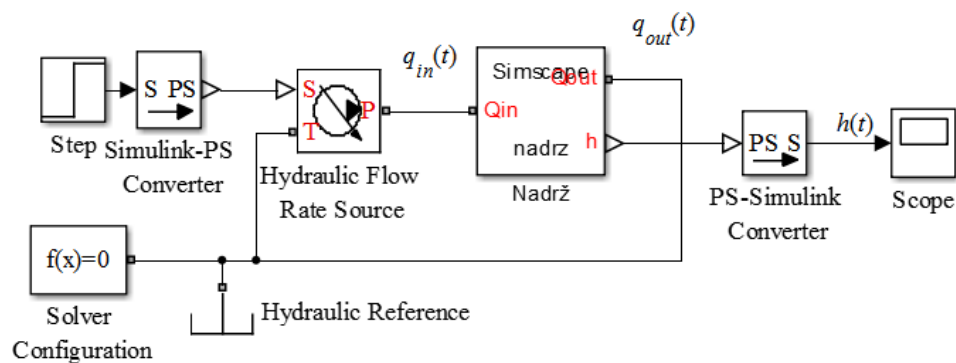
Rovnici pro hladinu si odvodíme za pomoci rovnice odtokové rychlosti jako

$$h = \left(\frac{q_{in}}{S_{odtok}} \right)^2 \frac{1}{2g} \quad (25)$$

Rovnice popisující odtok z nádrže potom bude

$$q_{out} = q_{in} - S_{nádrž} \frac{dh(t)}{dt} \quad (26)$$

Po zápisu těchto rovnic uložíme kód a přeložíme ho pomocí příkaz `ssc_build` „navez knihovny“. Otevřeme si vytvořenou knihovnu, použijeme vytvořený blok nádrže a vytvoříme zapojení s knihovnou Simscape.



Obr. 30. Nádrž – Simscape

V tomto případě je tvorba Simscape schématu o dost složitější protože bylo třeba vytvořit vlastní blok nádrže. Ale protože již tento blok máme, můžeme ho kombinovat s jinými součástmi hydraulické fyzikální oblasti a to je velkou výhodou přístupu Simscape.

Kód bloky nádrže (postup tvorby kódu viz. kapitola 3.2.3)

```
component nadrz
% Nadrz
nodes %vytvoření vstupu a výstupu v hydraulické doméně
    IN = foundation.hydraulic.hydraulic;% Qin:left
    OUT = foundation.hydraulic.hydraulic;% Qout:right
end
parameters % nastavitelné parametry bloku
    Sodtok = {1,'m^2'};
    Snadrz = {1,'m^2'};
    g={9.81,'m/s^2'};
end
outputs %výstup pro převod do simulink
    h = {0,'m'}; % h:right
end
variables %proměnné vstupů a výstupů
    qi = {0,'m^3/s'};
    qo = {0,'m^3/s'};
    pi = {0,'Pa'};
    po = {0,'Pa'};
end
function setup %podmínky pro zadávání parametrů modelu
    if (Sodtok <=0)
        error('Sv musí být větší než 0');
    end
    if (g <=0)
        error('g musí být větší než 0');
    end
    if (Snadrz <= 0)
        error('S musí být větší než 0');
    end
    across(pi, IN.p, []); %tlak na vstupu je zapsán do proměnné pi
    across(po, [], OUT.p); %tlak na výstupu je zapsán do proměnné po
    through(qi, IN.q, []); %přítok je zapsán do proměnné qi
    through(qo, [], OUT.q); %odtok je zapsán do proměnné qo
end
equations % rovnice popisující nádrž vztahy vstupů a výstupů
    h == qo^2/(2*g*Sodtok^2);
    qo == qi - h.der*Snadrz;
    po==pi;
end
end
```

2.4.3 Linearizace rovnic

Model nádrže není lineární, z důvodu přítomnosti odmocniny okamžité hladiny. Model je třeba linearizovat, linearizovaný model získáme tak že určíme pracovní bod, který bude počátkem v přírůstkovém modelu. Pracovní bod bude ve tvaru $[q_{in-0}; h_0]$. Nejdříve se zavedou přítoky a hladiny v přírůstkovém tvaru.

Tedy pro přítoky v přírůstkovém modelu bude platit

$$q_{in}(t) = q_{in-0} + \Delta q_{in}(t) \quad (27)$$

kde q_{in} původní průtok, q_{in-0} průtok odpovídající pracovnímu bodu, Δq_{in} přírůstek vzhledem k q_{in-0} a pro hladinu

$$h(t) = h_0 + \Delta h(t) \quad (28)$$

kde h_0 je hladina odpovídající pracovnímu bodu, $\Delta h(t)$ přírůstek vzhledem k h_0 .

Nyní se linearizuje hladina v přírůstkovém zápisu hladiny $\sqrt{h_0 + \Delta h(t)}$, pomocí prvních dvou členů Taylorova polynomu tak aby vznikla přímka tečná k statické charakteristice v pracovním bodě.

$$\sqrt{h(t)} = \sqrt{h_0 + \Delta h(t)} \approx \sqrt{h_0} + \frac{1}{1!} (\sqrt{h_0})' \Delta h(t) = \sqrt{h_0} + \frac{1}{2\sqrt{h_0}} \Delta h(t) \quad (29)$$

Po dosazení do diferenciálních rovnic kde jsou upraveny přítoky do přírůstkového zápisu, získáme rovnici

$$q_{in-0} + \Delta q_{in}(t) = S_v \sqrt{2g} \left(\sqrt{h_0} + \frac{1}{2\sqrt{h_0}} \Delta h(t) \right) + S(h_0 + \Delta h(t))'$$

Tuto rovnici lze výrazně zjednodušit ze znalosti počáteční hladina je nulová, tedy její derivace bude vždy nulová $h'_0 = 0$. Dále musí platit, že přítok a odtok jsou si v ustáleném stavu rovny, platí tedy následující rovnice

$$q_{in-0} = S_v \sqrt{2gh_0} \quad (30)$$

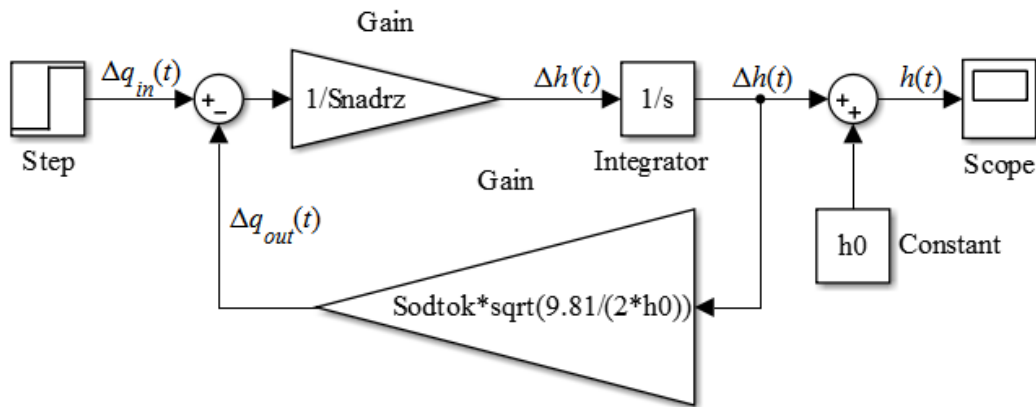
Proto lze tyto hodnoty odečíst a tím vznikne rovnici ve tvaru

$$\Delta q_{in}(t) = S_v \sqrt{\frac{g}{2h_0}} \Delta h(t) + S \Delta h'(t) \quad (31)$$

Vyjádřením nejvyšší derivace lze provést další simulaci v MATLAB/Simulink jako

$$\Delta h'(t) = \frac{\Delta q_{in}(t)}{S} - \frac{S_v}{S} \sqrt{\frac{g}{2h_0}} \Delta h(t) \quad (32)$$

a provede se ověření správnosti rovnice s pomocí základních bloků jako je *Sum*, *Gain* a *Integrator*. Signál povede z bloku *Step*, hladina pracovního bodu se získá pomocí bloku *Constant* a výsledné hodnoty se získají z bloku *Scope*.



Obr. 31. Nádrž – Linearizované diferenciální rovnice

2.4.4 Přenos linearizované nádrže

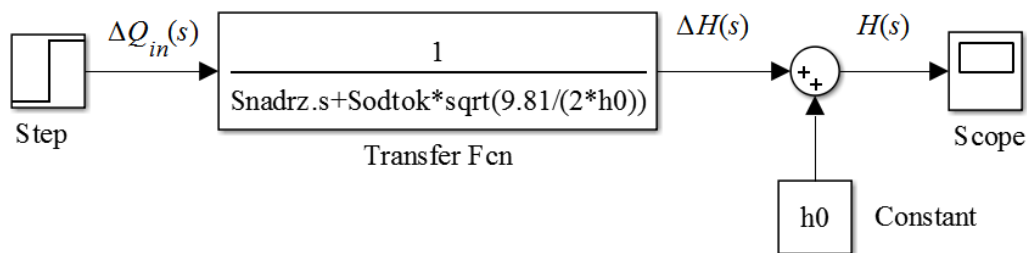
Po vytvoření linearizované diferenciální rovnice, je možné ji vyřešit pomocí Laplaceovy transformace. Po aplikaci zákona o n -té derivaci vznikne rovnice ve tvaru

$$\Delta Q_{in}(s) = S_v \sqrt{\frac{g}{2h_0}} \Delta H(s) + S \Delta H(s) \quad (33)$$

Přenos se odvodí jako poměr obrazů výstupu ku vstupu.

$$G(s) = \frac{\Delta H(s)}{\Delta Q_{in}(s)} = \frac{1}{Ss + S_v \sqrt{\frac{g}{2h_0}}} \quad (34)$$

Se znalostí přenosu je možné provést simulaci. Mezi hlavní použité bloky patří *Transfer Fcn* a *Constant*, který se přičte pomocí bloku *Sum*. Konstanta počáteční hladiny je přičtena k přírůstku, aby ve výsledku byla okamžitá hladina. Dále je připojen blok *Step* a na výstup *Scope* pro zobrazení grafu.



Obr. 32. Linearizovaná nádrž přenos

2.4.5 Stavový popis linearizované nádrže

Stavový popis určíme přímo z diferenciálních rovnic jednoduchou úvahou. Je zřejmé, že rozměry všech matic budou 1x1 protože rovnice je prvního řádu. Z tohoto důvodu stačí vyjádřit první derivaci přírůstku hladiny z rovnice, protože právě přírůstek hladiny bude stavovou proměnnou. Výstupní rovnice je zřejmá, protože cílem je zjistit přírůstek hladiny, proto výstupem bude přímo stavová proměnná.

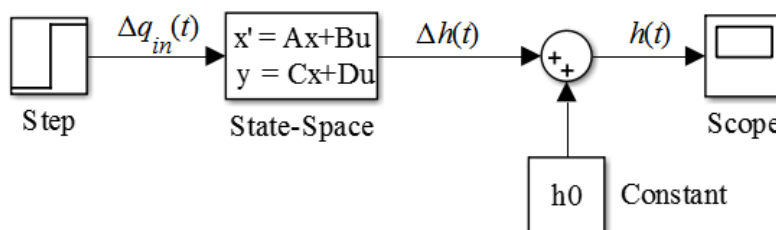
$$\Delta h'(t) = -\frac{S_v}{S} \sqrt{\frac{g}{2h_0}} \Delta h(t) + \frac{1}{S} \Delta q_{in}(t) \quad (35)$$

$$\Delta h(t) = 1 \Delta h(t) \quad (36)$$

Jednotlivé matice stavového popisu jsou potom

$$A = \left[-\frac{S_v}{S} \sqrt{\frac{g}{2h_0}} \right], B = \left[\frac{1}{S} \right], C = [1] \text{ a } D = [0]$$

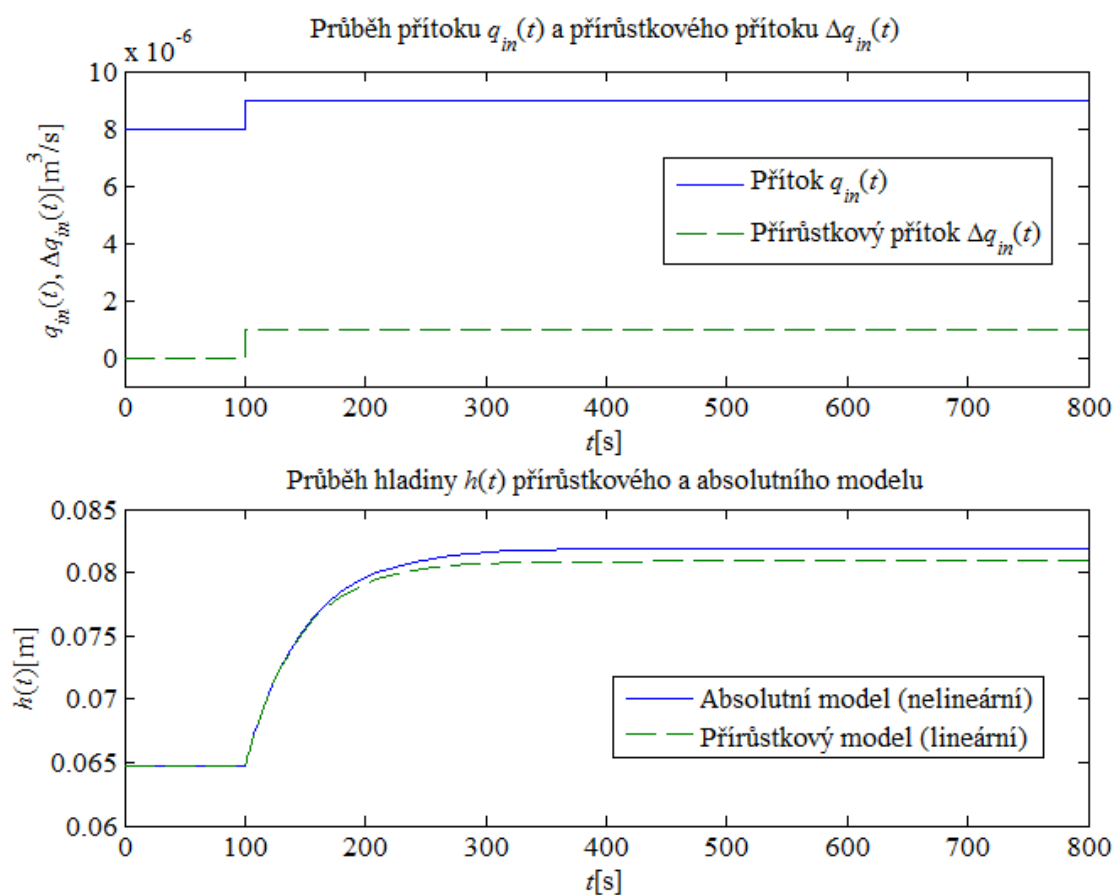
Se znalostí stavového popisu vytvoříme zapojení v MATLAB/Simulink s využitím bloku *State-Space*. Signál povede z bloku *Step*, hladina pracovního bodu se získá pomocí bloku *Constant* a výsledné hodnoty budeme číst z bloku *Scope*.



Obr. 33. Linearizovaná nádrž stavový popis

2.4.6 Porovnání simulací nádrže

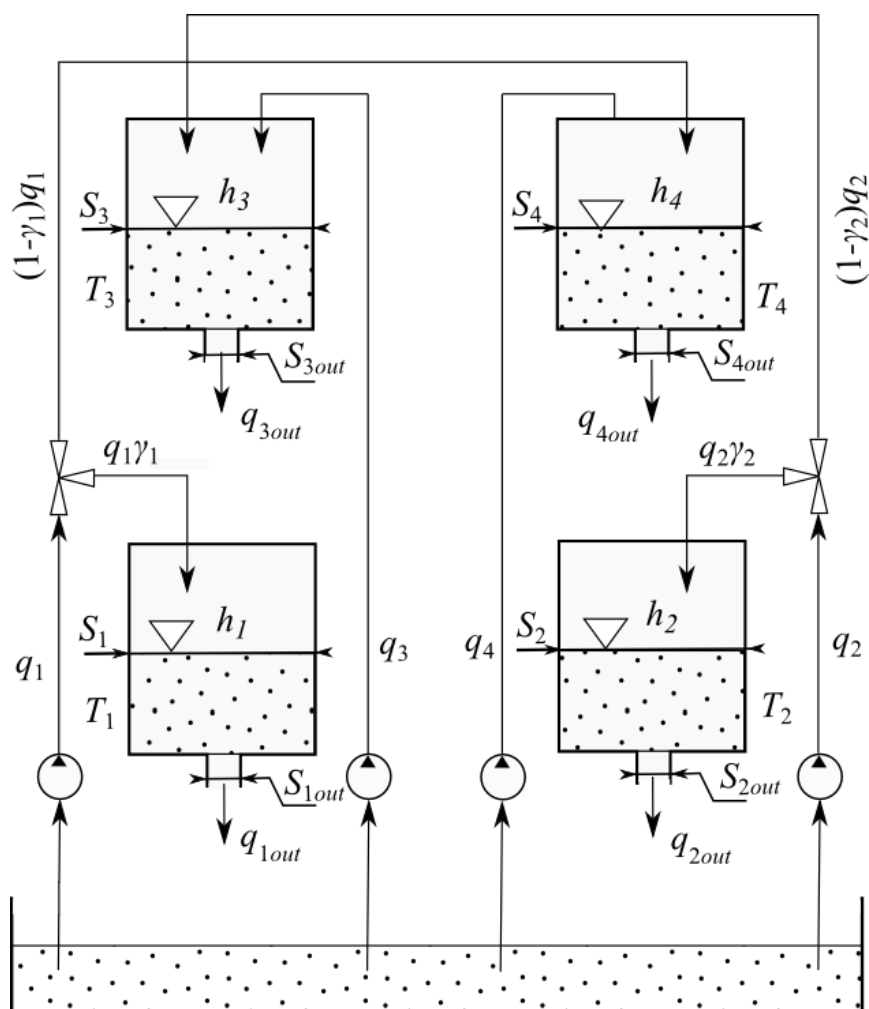
Nastavíme následné parametry simulace vstupní objemový průtok je reprezentován blokem *Step* a jeho velikost se v počátku změní z $q_{in}(t) = 0$ na $q_{in}(t) = 8 \cdot 10^{-6} \frac{\text{m}^3}{\text{s}}$. Plocha průřezu nádrže je $S_{nadrz}=0,0028\text{m}^2$ a plocha odtokového otvoru je $S_{odtok} = 7,1 \cdot 10^{-6}\text{m}^2$. Výsledkem budou stejné výstupy simulací u všech modelů. Jediný rozdíl bude u linearizovaných a nelinearizovaných modelů kde ve linearizovaných modelech dojde k linearizační chybě podle vzdálenosti od pracovního bodu.



Obr. 34. Porovnání výstupů simulací nádrže

2.4.7 Složitější hydraulický příklad

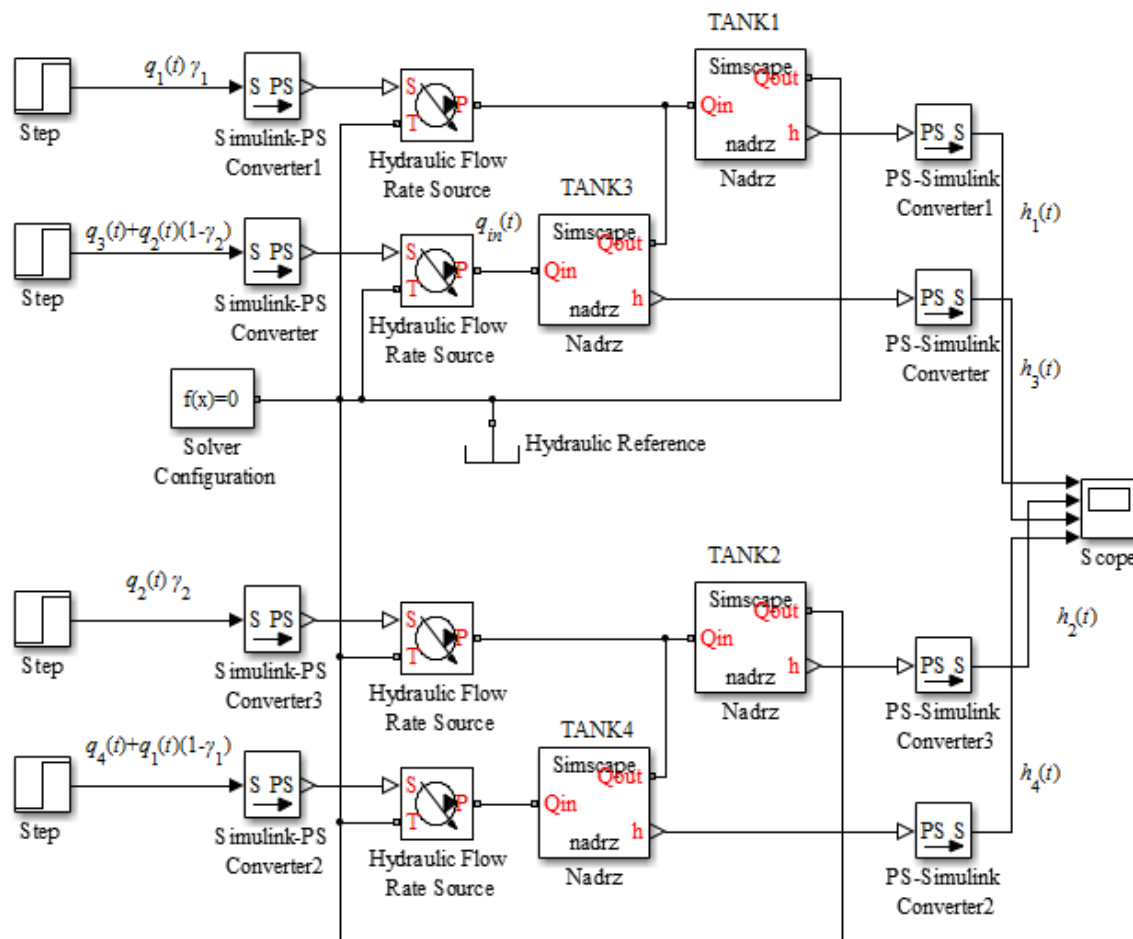
Jako složitější příklad byl zvolen systém s dvěma vstupy dvěma výstupy a dvěma poruchovými veličinami. Skládající se ze čtyř nádrží T_1, T_2, T_3 a T_4 , vždy s dvěma nádržemi nad sebou. Kohouty γ_1 a γ_2 dělíme přítok q_1, q_2 mezi dvě nádrže a to mezi levou spodní T_1 a pravou horní T_4 nebo pravou spodní T_2 a levou horní T_3 . Průřezy nádrží jsou označeny S_1, S_2, S_3 a S_4 , průřezy odtoků nádrží jsou označeny $S_{1out}, S_{2out}, S_{3out}$ a S_{4out} , jejichž indexy odpovídají číslům jednotlivých nádrží. Odtoky z jednotlivých nádrží jsou $q_{1out}, q_{2out}, q_{3out}$ a q_{4out} , kde čísla indexu odpovídají nádrži, ke které náleží daný odtok, dle [4]. Jedná se o MIMO systém, protože hladiny spodních nádrží budou závislé na hladinách horních nádrží. Dále je přidána poruchová veličina q_3 a q_4 , která působí pouze na horní nádrže.



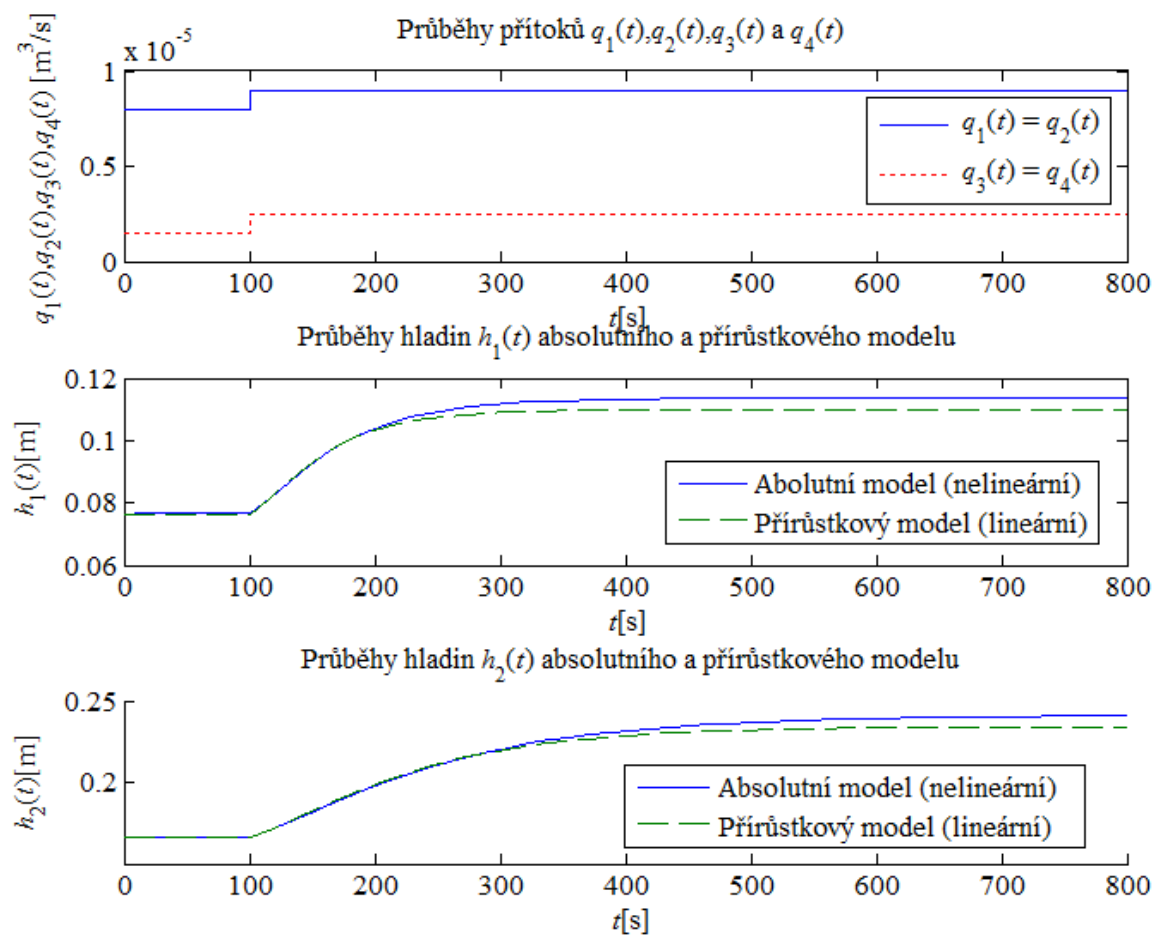
Obr. 35. Čtyřválcová vodárna

Pro zapojení je využito námi vytvořeného bloku z příkladu nádrže, tím lze současně ověřit modulárnost vytvořeného bloku. Dále jsou potřeba bloky *Hydraulic Flow Rate Source*,

Simulink-PS Converter, *PS-Simulink Converter*, *Hydraulic Reference* a *Solver Configuration*. Zdrojem signálu pro *Hydraulic Flow Rate Source* bude *Step* a výsledky se ověří pomocí bloku *Scope*.



Obr. 36. Zapojení vodárny s využitím knihovny Simscape



Obr. 37. Porovnání průběhů hladin čtyřválcové vodárny

3 JAZYK SIMSCAPE

Tento programovací jazyk umožňuje vytváření vlastních bloků a nových, nebo upravených fyzikálních oblastí. Program lze psát přímo v MATLAB nebo v jakémkoliv textovém editoru. Je nutno dát si pozor na velikost písmen protože tento jazyk je „case sensitiv“. Programovací jazyk je intuitivní a dovoluje používání některých matematických funkcí z MATLAB a jiné jsou specifické pro tento programovací jazyk. Struktura kódu se podobá objektovému jazykům. Jednotlivé části programu jsou vždy uloženy v programovém bloku, který označuje, co je v něm obsaženo např. `params`, „`inputs`“ a `equations`, každý programový blok je zakončen příkazem `end`. Hotový program je uložen s koncovkou „.ssc“ a je třeba ho přeložit v MATLAB. Po přeložení se vytvoří soubor s blokem, který jsme naprogramovali. Jak psát, editovat zdrojový kód a jak ho přeložit

Kód je vhodné psát v editoru MATLAB, to se provede výběrem „New → Script“, nebo klávesovou zkratkou Ctrl+N. To ovšem nestačí, protože MATLAB standardně nezvýrazňuje syntax jazyka Simscape, protože je rozdílný od objektově orientovaného jazyka MATLAB, proto je nutné nejdříve soubor uložit s koncovkou „.ssc“. Po uložení souboru se začne zvýrazňovat syntax „*Simscape language*“.

3.1.1 Nutná adresářová struktura a její překlad

Soubory Simscape musí být uloženy ve složce s názvem začínající znakem „+“ a nesmí obsahovat mezery či diakritiku, je také vhodné používat pouze malých písmen. Název této složky bude odpovídat názvu knihovny po překladu zdrojových kódů, která bude tvořena z bloků, jejichž zdrojové kódy jsou uloženy v této složce. Tyto složky mohou být součástí hierarchie, ale všechny složky obsahující kódy bloků musí začínat znakem „+“.

Po vytvoření kódu bloku a jeho uložení do správné složky, potřebujeme získat blok, který lze následně využít v Simulink proto je nutné se přepnout do nadřazené složky, která obsahuje nejvyšší složku naší hierarchie a provést příkaz `ssc_build +NazevSlozky`. Následně vám dá MATLAB na výběr z vašich nainstalovaných překladačů C/C++ pokud nemáte nainstalované žádné MATLAB vypíše chybu a je třeba nainstalovat kompatibilní překladač (více informací viz <http://www.mathworks.com/support/compilers/R2015a/index.html>). Po výběru překladače dojde k překladu všech kódů s koncovkou „.ssc“ uložených ve složkách začínajících znakem „+“. Tento překlad může trvat až několik minut na jeden blok.

Výsledkem překladu je soubor, který vznikne ve stejné složce, jako je uložena složka, na kterou byl volán příkaz `ssc_build`, soubor bude mít koncovku `„.mdl“` a za jeho jméno bude přidáno `„_lib“` a budou v něm Simscape bloky použitelné v MATLAB/Simulink.

Uvedeme si příklad, nejvyšší složka bude `„+elektrika“` a bude obsahovat složky `„+zdroje“`, `„+spotřebiče“` a složka `„zdroje“` bude obsahovat zdrojové kódy `„zdroj_proudu.scc“`, `„zdroj_napeti.scc“`. Pro překlad tedy půjdeme do složky, která je nadřazena složce `„+elektrika“` a na příkazovou řádku napíšeme `ssc_build +elektrika`. Vybereme překladač a to s čísly podle nabízených překladačů. Po překladu se nám ve složce nadřazené složce `„+elektrika“` vytvoří soubor `„Elektrika_lib.mdl“` který po otevření bude obsahovat výběr `„Zdroje“` nebo `„Spotřebiče“` po otevření složky zdroje se zobrazí dva bloky odpovídající našim zdrojovým kódům uloženým ve složce `„+zdroje“`. Tyto bloky lze přetáhnout do jakéhokoliv MATLAB/Simscape schématu, při přetažení je vytvořen link do této knihovny při změně kódu a jeho opětovném přeložení, bude v modelu umístěn vždy aktuální blok. Tento model je přenositelný, pouze pokud k němu dodáte vlastní vytvořenou knihovnu.

3.2 Syntax jazyka Simscape

Syntax jazyka Simscape je celkově velmi jednoduchý, jak již bylo zmíněno, zdrojový kód se tvoří z bloků kódu, které jsou zakončeny `end`. Tyto bloky jsou umístěny v rámci třídy definující fyzikální oblast `domain` nebo v rámci třídy definující komponentu `component`. Jazyk Simscape využívá dědičnost tříd, tedy jednotlivé třídy komponent dokáží dědit jiné třídy a to jak třídy `domain` tak třídy `component`. Lze také vytvořit nové komponenty, pouze zděděním jiných již vytvořených komponent, těmto třídám se říká kompozitní. Jednotlivé příkazy jsou vždy zakončeny středníkem. Stejně tak je důležité dodržovat velikost písmen, jazyk je tedy `„Case sensitive“`.

3.2.1 Bloky tříd

Bloky kódu jsou použity v rámci třídy `component` a `domain`. Třída `component` představuje jednu komponentu (např. pružinu), která dědí vlastnosti od třídy nebo několika tříd `domain` představující fyzikální oblast.

Syntax třídy `component` je následující

```
component nazev
    %Název bloku, který bude vyrendrován pod blokem v Simulink
    %Informace které budou vidět po rozklinutí bloku

    % Jednotlivé bloky kódu
end
```

Název komponenty musí odpovídat názvu souboru a nesmí obsahovat diakritiku a mezery. Název komponenty, který bude zobrazen v Simulink u bloku je napsán v prvním řádku poznámky a také nesmí obsahovat diakritiku, ale může mít mezery, v dalších řádcích je potom uveden text, který se zobrazí po rozkliknutí bloku v Simulink. Fyzikální oblasti, ze kterých se mají dědit proměnné jsou potom definovány v bloku `nodes` (viz 3.2.2). Podrobné informace naleznete v kapitole 3.4.

Rodičovská třída `domain` představující fyzikální oblast má syntax jako

```
domain nazev
    % Jednotlivé bloky kódu
end
```

Název domény musí odpovídat názvu souboru. A kód dědící tuto třídu musí obsahovat proměnné této fyzikální oblasti v bloku `variables`. Podrobné informace v kapitole 0.

3.2.2 Seznam bloků kódu pro deklaraci fyzikální oblasti nebo komponenty

V rámci jazyka Simscape existuje několik druhů bloků kódu, které zde budou uvedeny. Některé bloky jsou zakázány v rámci třídy `domain`, a dají se použít jen v rámci tříd `component`, vše je uvedeno v tabulce shrnující všechny bloky kódu.

Obecný předpis bloku je

```
TypBloku % Dále je možné uvést vlastnosti např. udávat, že blok je privátní atd.
    % Vlastní kód
end
```

Jednotlivé typy bloků pro deklaraci jsou v následující tabulce

Typ Bloku	Rodičovské třídy	Obsah bloku	Význam	Zapisovatelné
<code>parameters</code>	<code>domain</code> <code>component</code>	Číselná hodnota s jednotkou	Základní hodnoty měnitelných parametrů	Ano (po rozkliknutí bloku)
<code>variables</code>	<code>domain</code> <code>component</code>	Double hodnota s jednotkou	Základní počáteční podmínky	Ano (v kódu)
<code>inputs</code>	<code>component</code>	Double hodnoty s jednotkou	Vstupy ze Simulink	Ne
<code>outputs</code>	<code>component</code>	Double hodnoty s jednotkou	Výstupy ze Simulink	Ne
<code>nodes</code>	<code>component</code>	Instance rodičovských tříd <code>domain</code>	Definuje rodičovské třídy <code>domain</code> od kterých dědí jednotlivé porty	Ne

3.2.3 Vysvětlení jednotlivých bloků deklarace, postup tvorby kódu komponenty

Na začátek vytvoříme novou třídu `component` a pojmenujeme ji podle již popsaných pravidel, v rámci této třídy budeme postupně vytvářet jednotlivé bloky.

Jako ukázkový příklad v průběhu následujících kapitol, vytvoříme komponentu elektrického odporu, začínáme vytvořením následující třídy

```
component mujodpor
    %Odpor
    %Toto je tutorialový příklad s blokem odporu na využití jednotlivých
    %bloků kódu. V tomto bloku můžete zvolit hodnotu odporu R.
end
```

Na začátku tvorby kódu součásti začneme s blokem `nodes`, kterým se určí vstupy, výstupy a jejich fyzikální oblasti. Jedná se tedy o vytvoření instancí tříd `domain`. Jeden vstup nebo výstup je popsán jednou proměnnou, která se dědí vždy od jedné třídy `domain`. Odkaz na rodičovskou třídu se vytváří pomocí tečkové notace kde první je název nejvyšší složky přeložené knihovny a poslední je soubor obsahující zdrojový kód fyzikální oblasti. U těchto portů může být v Simscape vyrendrován popis, který je napsaný v poznámce a může být určena i poloha jednotlivých `nodes`. Obecně tedy jako

```
vstup=SložkováStrukturaKnihovny.KódFyzikálníOblasti; % Navez:Pozice
```

Mezi povolené příkazy označující pozice patří left (vlevo), right (vpravo), top (nahore), bottom (dole). Jednotlivé pořadí portů odpovídá vždy jejich inicializaci v bloku `nodes`.

Jako příklad budeme postupně vytvářet model elektrického odporu. Elektrický odpor má jeden vstup a jeden výstup. Využijeme doménu `electrical` definovanou v knihovně Simscape.

```
nodes
    vstup = foundation.electrical.electrical; % +:left
           %vstup jsme označili znaménkem + a bude vlevo
    vystup = foundation.electrical.electrical; % -:right
           %výstup jsme označili znaménkem - a bude vpravo

           %tyto notace odkazují na soubor
           %uložený v \+foundation\+electrical\electrical.ssc
end
```

Bloky `inputs` a `outputs` se skládají z inicializace jednotlivých vstupů nebo výstupů z/do Simulink v následujícím tvaru

```
proměnná={počáteční hodnota, 'jednotka'}; %Popisek
```

Jako druhý parametr je uvedena jakákoliv jednotka v případě bezrozměrného parametru je uvedeno `'1'`. Komentář `%Popisek` označuje slovo, které bude vyrendrované u vstupu/výstupu v Simulink. Zde daná jednotka, ve formě string (v apostrofech), by se měla nastavit v daném bloku převádějícím signál z/do Simulink.

V našem příkladu rezistoru zjevně nepotřebujeme ani jeden z těchto bloků ale pro ukázkou vyvedeme hodnotu právě nastaveného odporu, která by mohla sloužit pro další výpočty v Simulink. Stejně tak by mohla být přivedena hodnota odporu a nemusela by se v bloky nastavovat. Ukázka kódu s vývodem hodnoty odporu do simulink

```
outputs
    R = { 0, 'Ohm' }; % Hodnota odporu:top
           %Vytvoří výstup do simulink na horní straně bloku s vyrendrovaným
           %názvem „Hodnota odporu“
end
```

Bloky `parameters` a `variables` se skládají z inicializace jednotlivých proměnných ve stejném tvaru, jako `inputs/outputs`. Blok `parameters` obsahuje parametry, jejichž hodnotu bude možné nastavit po rozkliknutí bloku a v tomto okně budou jednotlivé parametry pojmenovány podle jejich komentáře `%Popisek`.

Naopak blok `variables` obsahuje pouze vnitřní proměnné, jejichž hodnotu lze nastavit pouze v kódu. Jednotku je důležité zapsat jako string, tedy do apostrofů. Jednotky v tomto případě nejsou jen tak na „okrasu“, ale mají zde významnou úlohu, u každého výpočtu se provádí odvození výsledné jednotky z dílčích jednotek, aby například nedošlo k tomu, že z výstupu na kterém chceme měřit napětí, změříme teplotu. Správnost jednotek se kontroluje při překladu kódu a porovnává se se signály úsilí a toku, v souboru fyzikální oblasti.

Bloky `parameters` a `variables` příkladového kódu elektrického odporu můžou vypadat následovně

```
parameters
    R = { 1, 'Ohm' }; % Odpor
    %Slovo Odpor bude zobrazeno jako název této proměnné pro rozkliknutí bloku
end
variables
    i = { 0, 'A' };
    v = { 0, 'V' };
    % Zavedení proměnných definovaných ve fyzikální oblasti
end
```

V rámci této kapitoly byl vytvořen následující kód

```
component mujodpor
    %Odpor
    %Toto je tutorialový příklad s blokem odporu na využití jednotlivých
    %bloků kódu. V tomto bloku můžete zvolit hodnotu odporu R.
    nodes
        vstup = foundation.electrical.electrical; % +:left
            %vstup jsme označili znaménkem + a bude vlevo
        vystup = foundation.electrical.electrical; % -:right
            %výstup jsme označili znaménkem - a bude vpravo

        %tyto notace odkazují na soubor
        %uložený v \+foundation\+electrical\electrical.ssc
    end
    outputs
        R = { 0, 'Ohm' }; % Hodnota odporu:top
        %Vytvoří výstup do simulink na horní straně bloku s vyrendrovaným
        %názvem „Hodnota odporu“
    end
    parameters
        R = { 1, 'Ohm' }; % Odpor
        %Slovo Odpor bude zobrazeno jako název této proměnné pro rozkliknutí bloku
    end
    variables
        i = { 0, 'A' };
        v = { 0, 'V' };
        % Zavedení proměnných definovaných ve fyzikální oblasti
    end
end
```

Můžeme si všimnout, že zatím chybí zásadní věci jako je rovnice popisující prvek a propojení vstupů a výstupů bloku s jednotlivými proměnnými třídy fyzikální domény. Toto je doplněno v následujících dvou kapitolách a to z důvodu že následující bloky jsou komplikovanější.

3.2.4 Blok MATLAB funkce „function setup“

V rámci kódu `component` je navíc k dispozici provedení jedné MATLAB funkce, která slouží pro ověření správnosti zadávacích polí, jejich omezení nebo návratu chybových hlášek. Jednou z nejdůležitějších činností co tato funkce vykonává, je přiřazení jednotlivých prvků bloku `nodes` proměnným domény. Následně může sloužit pro práci s kompozitními komponentami.

Tato funkce má jméno `setup` a do Simscape kódu se zapisuje následujícím způsobem

```
function setup
% Vlastní kód v jazyku MATLAB
% Podmínky vstupních parametrů
%Propojení proměnných fyzikální oblasti a finálního bloku
end
```

Jméno funkce `setup` je neměnné a funkce nesmí přijímat ani vracet žádné proměnné. Tato funkce je volána jen jednou a to při kompilaci modelu v MATLAB/Simulink před provedením simulace. V bloku `function` lze provádět i nějaké výpočty, ale musíme mít na paměti, že tyto výpočty budou provedeny jen jednou a to při startu simulace, toho se využívá pouze ve speciálním případě, když nepoužíváme blok `equations` (více viz 3.4.2). Také je zde možné definovat počáteční podmínku z nastavitelného parametru (např. v elektrickém systému máme parametr `i0` a proměnnou `i` tak v tomto bloku vložíme výraz `i=i0`; tím definujeme počáteční podmínku proudu).

Primárním účelem tohoto bloku je tedy přiřazení proměnných toku a úsilí domény k vstupům Simscape bloku, toho je dosaženo pomocí dvou funkcí. Funkce `across` definuje veličinu toku a funkce `tough` definuje veličinu úsilí. Je důležité si uvědomit, že tímto definujeme také kladný směr a záporný směr signálu v bloku. Obecný tvar těchto funkcí je stejný a vypadá následovně

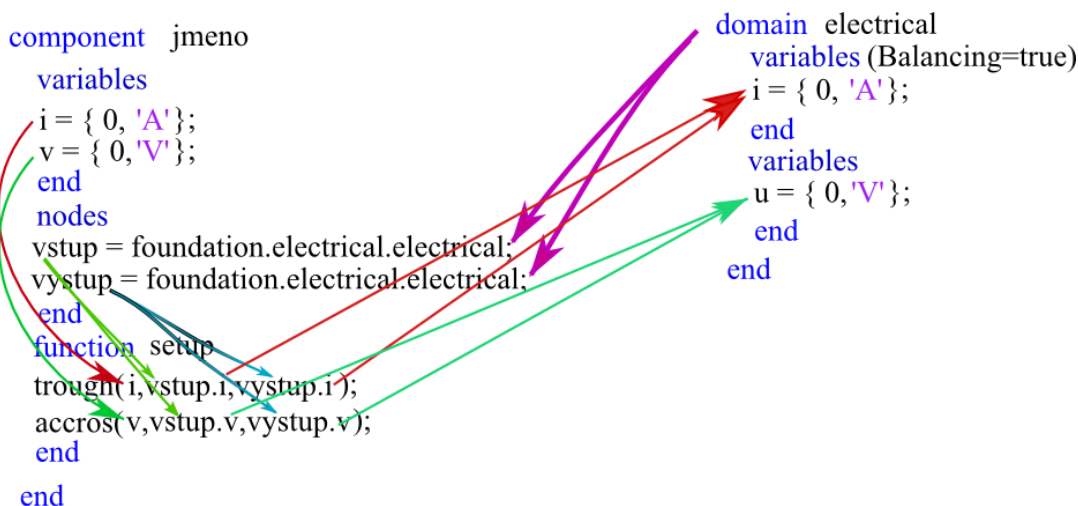
```
tough(PromenaVariables, NodeVstup.OdkazNaPromenouDoFyzikalniOblasti,
NodeVystup.OdkazNaPromenouDoFyzikalniOblasti);
```

Jedná se tedy o přiřazení jedné proměnné z bloku `variable`, instancím fyzikálních oblastí vstupu a výstupu z bloku `nodes` u kterých je odkazováno na danou proměnnou fyzikální oblasti z `domain` pomocí tečkové notace. Jedna z možností je také přivést veličiny na referenční zem, protože je nechceme na výstupu, to se provede vložením uzavřených hranatých závorek `[]` místo odkazu na instanci třídy `domain` z bloku `nodes`.

Pro náš příklad rezistoru by tyto funkce mohly vypadat následovně

```
trough(i,vstup.i,vystup.i);    %Veličina úsilí
% První i značí proměnou z bloku variable v komponentě mujodpor
% Význam vstup.i je že naši proměnné i v bloku mujodpor přiřadíme instanci třídy
% domain z bloku node ale tato třída neví ke které proměnné i z našeho bloku
% mujodpor přiřadit tuto proměnnou, proto pomocí tečkové notace vybereme
% proměnnou proudu tedy i
accros(v,vstup.v,vystup.v);    %veličina toku
% Význam zde je úplně stejný proměnnou v bloku mujodpor přiřazujeme vstupy,
% výstupy a zároveň propojujeme proměnou domény s proměnou našeho bloku
```

Je možné, že výše uvedený příklad nemusí každý pochopit, proto zde uvádím schéma, z kterého je zřejmé co a jak na sebe navazuje.



Obr. 38. Vzájemné vztahy proměnných

Další povolenou činností v rámci tohoto bloku je ošetření rozsahu proměnných, pomocí podmínky `if`. Jedná se o poměrně standardní formát zápisu této podmínky jako

```
if podmínka
    %vykonávaný kód
elseif podmínka
    %vykonávaný kód
else
    %vykonávaný kód
end
```

Rozhodovací znaky podmínek použitelné v tomto bloku

==	Rovnost (neplést se znakem = který symbolizuje přiřazení)
~=	Nerovnost
<	Menší než
>	Větší než
>=	Větší nebo rovno
<=	Menší nebo rovno
~	Negace výrazu

Tab. 19. Rozhodovací výrazy v podmínkách

V podmínkovém ošetření je možné použít chybové hlášky, která zobrazí chybu parametru podle заданých podmínek a zastaví průběh simulace. Toto se provádí pomocná funkce `pm_error` jejíž tvar je následující

```
pm_error('simscape:TypRovnostiNerovnosti','NázevChybnehoParam')
%První parametr této funkce má za význam, ukázat že došlo k chybě v Simscape
%a typ nerovnosti, který byl porušen
%Druhý parametr je název jednotky a dodatečný popis
```

Ošetření zadatelného parametru našeho bloku rezistoru by mohlo vypadat následovně

```
if R<0
    pm_error('simscape:GreaterThanOrEqualToZero','Odpor nesmí byt menší než 0')
end
```

Ošetřujeme tedy, aby odpor nebyl menší než nula.

3.2.5 Blok rovnic, „equations“

Blok kódu `equations` je nejčastěji samotným jádrem Simscape bloku které obsahuje rovnice popisující vztahy vstupů a výstupů. Je ve tvaru standartního zápisu tedy na počátku `equations` a na konci `end`. V bloku `equations` platí několik následujících pravidel a možností:

- Musí obsahovat správný počet rovnic. Uvedeno na příkladu v elektrické fyzikální oblasti, můžeme mít jednu rovnici popisující napětí, jednu rovnici popisující proud, nebo obě dohromady. V případě, když je uvedena jen jedna rovnice druhá veličina se nemění. Dále může mít rovnice popisující vnitřní proměnné bloku. Tento blok, ale nikdy nesmí obsahovat více rovnic popisujících stejnou veličinu, jinak při simulaci s tímto blokem dojde k chybě. (nesmíme `v==v1;v==v2;`) To je z důvodu, že všechny rovnice výsledného modelu jsou řešeny jako soustava a ta by neměla žádné řešení.
- Jednotka výsledku rovnice bude odpovídat jednotkám účastníků rovnice. Zároveň jednotky výsledků musí odpovídat definici jejich proměnných v bloku `variable`.
- Rovnice smí obsahovat pouze znaménko rovnosti (znaménko rovnosti je `==`) a ne přiřazení (znaménko přiřazení je `=`).
- V žádné rovnici nesmí být proměnná rovna číslu (nesmíme `x==5`), vždy se proměnné musí rovnat jiným proměnným (můžeme `x==x1`). To je z důvodu zachování fyzikální realističnosti. Toto pravidlo má jednu výjimku a to je definice referenčního bodu kdy může být obsažen výraz roven nule (např. `u==0`).
- V případě větvení je možné použít podmínku se stejným předpisem jako v předchozí kapitole. Podmínky je možné používat i v jednotlivých rovnicích ale zase platí, že v žádné větvi se proměnná nesmí rovnat číslu vždy pouze proměnné.

```
equations
    x==if 5>0, x1 else x2 end;    %Podmínka v rovnici
end
```

- Ve funkcích je možné využívat Simulink proměnnou `time`. (např. `x==sin(w*time)`)

- Jako rovnice je často třeba využít diferenciální rovnice proto je zde možnost derivace proměnné. Ta se provede pomocí tečkové notace za proměnnou a příkazem `der`. Jde tedy provést pouze první derivaci, pro vyšší derivace musíme vytvořit další rovnice popisující tyto derivace.

```
equations
    x1==y.der;    %První derivace y
    x2==x1.der;   %Druhá derivace y
end
```

Mezi další a pokročilé vlastnosti bloku `equations` patří následující:

- Pro zpřehlednění je možné využít v bloku `equations` další blok `let in`, v kterém budou umístěny proměnné substituované do rovnic, jako přiřazení novým proměnným ne jako rovnice. Tyto substituované hodnoty lze použít do následujícího bloku `end`. Substituovaná proměnná nemusí být konstantní. A je zde možné využít podmínkových výrazů.

```
equations
    let
        v=v1+v2;    %Vytvoření substituované části rovnice
        v= if 1<0, v1 else v2 end; %ukázková podmínka
        [v1,v2]= if 2>0, v1; v2 else v3; v4 end; %Podmínka s výsledkem vektoru
    in
        x==v.der;    %Využití substituované části rovnice
    end
end
```

- Další výhodnou možností bloku `equations` je využití tabulkových hodnot, které jsou zadavatelné jako vektor po rozkliknutí bloku v Simscape. Tohoto se dá využít u různých tabulkově se chovajících systémů, které se nedají pospat rovnicemi. Tyto tabulkové hodnoty se ukládají do bloku `parametres` s vlastností `Size=variable`. Práce s těmito hodnotami se provádí pomocí funkce `tablelookup`, která vrací interpolovanou hodnotu z daných vektorů a je v následujícím tvaru

```
tablelookup(x1d, x2d, yd, x1, x2, interpolation = linear|cubic|spline, extrapolation = linear|nearest)
```

kde `x1d` je jeden směr a `x2d` je druhý směr (pouze pro 2d) `yd` je vyhledávaná hodnota v tabulce, `x1` je vstupní hodnota v jednom směru a `x2` ve druhém směru. Interpolace umožňuje výběr jejího typu a dále extrapolace pokud je hodnota mimo

rozsah tabulky. Parametry pro druhý rozměr ani definice interpolace a extrapolace nejsou povinné.

Jedná se tedy o přiřazení tabulkových hodnot vstupu x1d okamžitému vstupu x1 a potom jejich následnému převedení odpovídající hodnotě výstupu yd. Pokud nalezneme přesnou hodnotu odpovídající okamžitému vstupu v tabulce tak je vrácena funkcí pokud ovšem není tato hodnota tabulace, tak se provede interpolace nebo extrapolace, podle hodnot tabulky a je vrácena takto získaná hodnota.

Možná ukázka využití funkce tablelookup

```
inputs
    u = 0;
end
outputs
    y = 0;
end
parameters (Size=variable) %Definice vlastnosti pro tabulkové vyhledávání
    xd = {[100 200 300 400] 'K'};
    %Tabulka z které vybereme podle vstupu
    yd = {[1e5 2e5 3e5 4e5] 'Pa'};
    %Tabulka z které vybereme výstup podle předchozí tabulky a druhu interpolace
end
equations
    y == tablelookup(xd, yd, u, interpolation=linear);
    %Pokud tedy na vstup přivedeme signál o 100K potom na výstupu y podle
    % tabulky dostaneme 1e5Pa
end
```

- Dále je možné testovat jednotlivé veličiny na jejich hodnoty a podle nich vracet chyby v průběhu simulace. Tohoto se dá vhodně využít např. při překročení bezpečných hodnot daného bloku. Toto se dá provést přidáním funkce assert. Standardní zápis této funkce je následující

```
assert(PodmínkaSprávnosti, 'NavracenaHlaska');
```

Pokud bude porušena podmínka správnosti, zobrazí se chybová hláška v okně Simulink.

Ukázka možnosti ošetření dynamického parametru R

```
equations
    assert(R >=0, 'Hodnota odporu nesmí klesnout pod nulu');
    % V případě NESPLNĚNÍ podmínky se zobrazí tato chybová hláška
end
```

- V určitých operacích je třeba převést rozměrnou proměnnou na bezrozměrnou a to nejčastěji v určitých případech číselných exponentů, z důvodu, že by tento exponent byl uplatněn i na jednotku. Dále toto může nastat u násobení bezrozměrnou konstantou, kdy se ale mění finální jednotka.

Toto lze provést pomocí funkce `value`, která odebere jednotku proměnné. Následně musíme přiřadit jednotky, tak aby byl výsledek správně a to opět pomocí funkce `value` nebo přiřazením jednotky celé bezrozměrné straně rovnice. Funkci `value` je také možné použít pro převádění jednotek fyzikálních domén (např. stupně Celsia na Kelvina, metry na cm a podobně).

```
parameters
k={ 0.356, '1'}; %bezrozměrný parametr získaný experimentálně
end
variables
p={ 0, 'Pa'}; %Tlak
q={ 0, 'm^3/s'}; %Prutok
end
equations
p == k * q^1.023; %výsledkem této formulace je nesprávný a to i přesto že pomocí
%experimentálně získané konstanty k víme že to takto skutečně je, protože
%jednotky nesouhlasí(Pa neodpovídá m^3/s).
p == { k * value(q, 'm^3/s')^1.023, 'Pa' }; %Toto je jediný správný zápis
% v Simscape kódu. Musíme nejprve odebrat jednotku m^3/s od proměnné q a
%posléze přidělit výslednou jednotku celé bezrozměrné straně aby u jednotek
%platilo, že Pa na jedné straně odpovídá Pa na druhé
end
```

Tedy pro náš příklad odporu definujeme rovnici pro napětí pomocí Ohmova zákona v následujícím tvaru.

```
equations
v==R*i ;
end
```

Po doplnění bloku `equations` a propojení domény a bloku ve `function` setup je blok odporu hotový a připraven k použití.

```
component mujodpor
    %Odpor
    %Toto je tutorialový příklad s blokem odporu na využití jednotlivých
    %bloků kódu. V tomto bloku můžete zvolit hodnotu odporu R.
    nodes
        vstup = foundation.electrical.electrical; % +:left
            %vstup jsme označili znaménkem + a bude vlevo
        vystup = foundation.electrical.electrical; % -:right
            %výstup jsme označili znaménkem - a bude vpravo

        %tyto notace odkazují na soubor
        %uložený v \+foundation\+electrical\electrical.ssc
    end
    outputs
        R = { 0, 'Ohm' }; % Hodnota odporu:top
        %Vytvoří výstup do simulink na horní straně bloku s vyrendrovaným
        %názvem „Hodnota odporu“
    end
    parameters
        R = { 1, 'Ohm' }; % Odpor
        %Slovo Odpor bude zobrazeno jako název této proměnné pro rozkliknutí bloku
    end
    variables
        i = { 0, 'A' };
        v = { 0, 'V' };
        % Zavedení proměnných definovaných ve fyzikální oblasti
    end
    function setup
        if R<0
            pm_error('simscape:GreaterThanOrEqualToZero','Odpor nesmí být menší než 0')
        end
        trough(i,vstup.i,vystup.i);
        accros(v,vstup.v,vystup.v);
    end
    equations
        v==R*i ;
    end
end
```


3.3 Fyzikální oblast „Domain“

Pro všechny fyzikální oblasti v Simscape musí být aplikovatelná veličinová analogie pomocí úsilí a toku. Každý blok ať už je součástí již stávajících knihoven nebo nově vytvořený musí být součástí nějaké fyzikální oblasti, nebo alespoň jeho jednotlivé vstupy musí být přiřazeny různým fyzikálním oblastem. V této kapitole je popsána struktura souboru fyzikální oblasti.

3.3.1 Struktura souboru fyzikální oblasti

Kód fyzikální oblasti je napsán v bloku `domain`, který obsahuje proměnnou, která bude úsilím (měříme paralelně k prvku) a tokem (měříme sériově s prvkem). Toto je definováno ve dvou blocích v bloku `variables` je uvedeno úsilí a v bloku `variables` (Balancing = true) je uveden tok. Dále může být uveden blok `parameters` s globálně nastavenými parametry sdílenými mezi všechny bloky. Blok `variables` (Balancing = true) je povinný a bez jeho přítomnosti vrátí překladač chybu.

V jednotlivých blocích `variables` definujících fyzikální oblast je možné uvést více proměnných, např. je možné vytvořit systém, který bude hydraulický, ale zároveň bude kapalina elektricky vodivá. Budou tedy dvě veličiny toku proud, průtok a dvě veličiny úsilí napětí a tlak.

V blocích `variables` jsou uvedeny počáteční hodnoty všech prvků a jejich jednotky ve tvaru

proměnná={PočátečníHodnota, 'jednotka'}

Ve stejném tvaru jsou i hodnoty globálních parametrů v bloku `parameters`.

Ukázka jak může vypadat definice elektrické domény

```
domain elektrika    % Název domény shodný s názvem souboru
parameters          %Globální parametry (nepovinný blok)
    Temperature = { 300.15 , 'K'    }; % Teplota obvodu
    GMIN        = { 1e-12 , '1/Ohm' }; % Minimální vodivost
end
variables            % Proměnné úsilí
    v = { 0 , 'V' };
end
variables(Balancing = true) % Proměnné toku
    i = { 0 , 'A' };
end
end
```

Tento kód je možné zapsat před začátek definice komponenty a přímo se na něho odkazovat jeho jménem tedy „elektrika“, nebo do samotného souboru s koncovkou „.ssc“, potom je nutné se odkazovat pomocí tečkové notace.

Příkladem může být složka „+elektrika“ ve které je uložen soubor „elektrika.ssc“ definující elektrickou fyzikální oblast. Pokud budeme chtít vytvořit blok v této fyzikální oblasti tak to provedeme pomocí tečkové notace jako „elektrika.elektrika“, tedy obecně „CestaKSouboru.SouborDefinujícíFyzikálníOblast“. Soubor s fyzikální oblastí musí být uložen v hierarchii překládané knihovny.

3.3.2 Globálně nastavené parametry vs. rozlévání parametrů

Jak bylo naznačeno v předchozí kapitole, mohou existovat nějaké globální parametry. V elektrickém systému, to může být například teplota, která bude ovlivňovat odpor a jiné vlastnosti elektrických součástí. Existují dvě možnosti definice globálních proměnných první je definice v rámci souboru fyzikální oblasti v bloku `parameters`, v tomto případě bude tento parametr společný pro všechny bloky obsahující v `nodes` instance třídy této fyzikální oblasti. Nevýhodou této volby je, že tyto globální parametry lze měnit jen v kódu, nebo je třeba vytvořit jiných parametrů a jejich hodnotu potom přiřadit globálnímu parametru.

Druhou možností je rozlévání parametrů z jednotlivých komponent do jiných. Tato vlastnost je nastavitelná pro všechny komponenty a důsledkem bude, že komponenty připojené ke komponentě s touto vlastností budou mít přístup k bloku kódu `parameters` této komponenty. To znamená, že nastavením parametru v komponentě s touto vlastností ji nastavím i všem, které jsou k ní připojené. Tato vlastnost je definována u bloku kódu `component` jako `Propagation = source`.

```
component ( Propagation = source ) odpor
  parameters
    T= { 300.15 , 'K' }; %Teplota
  end
end
```

K tomuto parametru potom přistoupíme z jiného bloku pomocí tečkové notace následujícím způsobem

```
component odpor2
    nodes
        a = foundation.electrical.electrical;
        % Tento port bude připojen k předchozímu bloku
    end
    equations
        t==a.T;
        %Přístup pomocí tečkové notace pro hodnotu teploty T z předchozího bloku
    end
end
```

3.4 Komponenty „*component*“

Tato kapitola pojednává o možnostech vytvoření nové komponenty. Ten nejjednodušší způsob byl předveden v kapitolách 3.2.3, 3.2.4, 3.2.5 a dále se mu nebudu věnovat. Jeho principem byl nejlogičtější přístup a to sice vytvoření vstupů, výstupů, proměnných, parametrů, portů a rovnic. Tento přístup není ovšem jediný Simscape nabízí ještě dva přístupy, prvním je zdědění základních částí komponenty a tím vytvoření podtřídy k jiné třídě komponenty. Tuto možnost lze využít například při vytváření velkého množství podobných komponentů, které jsou popsány různými rovnicemi.

Další možností je tvorba kompozitní komponenty, která využívá velké výhody Simscape a sice tu že nemusíme znát matematické modely celku, ale stačí nám znát pouze modely jednotlivých komponent nebo dokonce neznat ani ty a využít už předem vytvořených Simscape bloků. Kompozitní komponenta je tedy složena z jiných již vytvořených komponent komponent. Některé možná napadne, jestli tento způsob má nějaký význam, když si již vytvořené bloky můžeme složit v Simulink a nemusíme vytvářet další komponenty. Význam to má určitě veliký, protože tímto způsobem dokážeme zjednodušit již existující modely v Simscape nebo vytvořit nové složitější například komponentu motoru vytvoříme pomocí rezistorů, cívek z elektrických komponent potom elektro mechanického rotačního převodníku z elektrických rotačních prvků a z mechanických prvků y se využilo tření a setrvačnost hmoty, vytvoření instancí těchto komponent v kódu kompozitní komponenty a jejich následným propojením získáme blok motoru podle našich požadavků. Tímto způsobem dokážeme velmi jednoduše přidávat nebo oddělovat další komponenty bez nutnosti opakovaných výpočtů matematického modelu.

3.4.1 Vytvoření podtřídy komponenty zděděním vlastností rodičovské třídy

Tuto metodu je vhodné aplikovat na součásti stejné fyzikální oblasti, a které jsou si velmi podobné. Jedná se vlastně o vytvoření rodičovské třídy a zdědění jejich vlastností do tříd jednotlivých komponent.

Ukázkově si zde definujeme rodičovskou třídu `component`. Tato třída sama o sobě nemá žádný význam, je to jen jakýsi mustr, kdyby tato třída byla přeložena tak by vznikl blok se vstupem a výstupem, který by nic nedělal, to co by do něj vstoupilo, to by i vyšlo a neměl by žádné vlastnosti. Ve většině případů tento blok nebudeme chtít samostatně přeložit proto u `component` přidáváme vlastnost `Hidden=true`, která zabrání ve vytvoření tohoto bloku.

```
component(Hidden=true) elrodic %Blok nebude samostatně přeložen
nodes
  vstup = elektrika.elektrika; % +:left
  vystup = elektrika.elektrika; % -:right
end
variables
  i = { 0, 'A' };
  v = { 0, 'V' };
end
function setup
  through( i, vstup.i, vystup.i );
  across( v, vstup.v, vystup.v );
end
end
```

Tuto rodičovskou třídu komponenty lze zdědit do jakéhokoliv bloku komponenty ale podmínkou je že každý komponent může mít pouze jednoho rodiče. Zdědění se provádí pomocí znaku „<“ vedle názvu komponenty, opět pomocí tečkové notace odkazující na soubor rodičovské třídy. V tomto případě je rodičovská třída „elrodic.ssc“ třída další generace je „mujodpor.ssc“ oba tyto soubory jsou uloženy v hlavní složce knihovny elektrika „+elektrika“ stejně jako soubor fyzikální oblasti „elektrika.ssc“.

Ukázka zdědění předcházející rodičovské třídy, kdy výsledkem bude rezistor se vstupem a výstupem ve fyzikální oblasti elektrika a parametrem odporu.

```
component mujodpor < elektrika.elrodic %Zdědění rodičovské třídy
parameters
    R = { 1, 'Ohm' }; %Odpor
end
function setup
    if R < 0, pm_error('simscape:GreaterThanOrEqualToZero','Odpor') end;
end
equations
    v == R*i;
end
end
```

Je tedy zřejmé, že jsme ušetřili relativně velké množství kódu a zde vytvořenou rodičovskou třídu se vstupem a výstupem v elektrické fyzikální oblasti lze aplikovat na velkou část elektrických součástí (např. cívky, kondenzátory, diody ...).

3.4.2 Kompozitní komponent, vytvoření instance třídy komponenty

Kompozitní komponent se zásadně liší od jiných komponentů a to tím, že není tvořen matematickými vztahy, neobsahuje tedy blok `equations`, ale jinými komponenty. Instance komponent se vytváří v bloku kódu `components`, dále je třeba tyto komponenty propojit k tomu slouží blok `connection`.

Blok `components` by měl mít vlastnost `Hidden=true` aby se zamezilo opakovanému přeložení a vytvoření bloků komponent v bloku kódu `components`. Každá instance komponenty musí mít přiřazené jméno a cesta k ní je vytvořena pomocí tečkové notace. Počet stejných nebo celkových komponent není nijak omezen. Každá komponenta může přijímat parametry pro každý jeden parametr v instanciované komponentě. Tento parametr se zadává do kulatých závorek za instanciovanou komponentu jako parametr instanciované komponenty, kterému je přiřazen parametr této kompozitní komponenty.

Ukázka využití bloku `components` a předávání parametrů instancím komponent

```
parameters
  R2param={ 1, 'Ohm' }; %Odpor
end
components (Hidden=true)
  R1 = foundation.electrical.electrical.resistor; %První instance komponenty resistor
  R2 = foundation.electrical.electrical.resistor(R=R2param);
  %Druhá instance komponenty resistor
  %Parametr R1param nastavený po rozkliknutí bloku je přiřazen proměnné R v
  %instanci
  I = foundation.electrical.electrical.inductor; %Instance komponenty inductor
end
```

Blok kódu `connections` může obsahovat pouze funkce `connect` jejichž účelem je propojit jednotlivé porty komponentů do žádané podoby výsledného obvodu a hlavně tyto komponenty napojit na porty kompozitní komponenty. Obecný zápis této funkce je následující

```
connect(Port1, Port2); %Tímto zápisem propojíme dva porty Port1 a Port2
connect(Port1, Port2, Port3);
%Tímto zápisem propojíme tři porty Port1 a Port2 a Port3
```

Funkce `connect` je definovaná pro dva nebo tři parametry, v případě že potřebujeme více propojení, použijeme další příkaz `connect`. Počet těchto propojení není nijak omezen a není třeba dodržovat nějaké speciální pořadí parametrů této funkce.

Obecně propojování bývá složitější, protože je potřeba pomocí tečkové notace přistupovat do instance komponent do bloku kódu `nodes` pro jednotlivé porty. Proto je zde uveden jednoduchý příklad bude se jednat o paralelní propojení dvou rezistorů z existujících bloků dostupných z knihovny Simscape.

```
component paralelresistor
    nodes
        vstup= foundation.electrical.electrical;
        vystup= foundation.electrical.electrical;
    end
    parameters
        R1param={ 1, 'Ohm' }; %Odpor
        R2param={ 1, 'Ohm' }; %Odpor
    end
    components (Hidden=true)
        R1 = foundation.electrical.electrical.resistor(R=R2param);
        R2 = foundation.electrical.electrical.resistor(R=R2param);
        %Tyto komponenty mají definovaný blok nodes jako
        %   nodes
        %       p = foundation.electrical.electrical; % +:left
        %       n = foundation.electrical.electrical; % -:right
        %   end
        % kde p je vstup a n je výstup z komponenty
    end
    connections
        connect(vstup,R1.p,R2.p)
        %Propojení vstupu do kompozitního bloku se vstupy do rezistorů R1 a R2
        connect(vystup,R1.n,R2.n)
        %Propojení výstupu do kompozitního bloku s výstupy do rezistorů R1 a R2
    end
end
```

V této kapitole uvedeme ještě jednu ukázkou, a sice ukázkou kompozitního integračního článku s integrovaným voltmetrem. Budeme mít dva elektrické porty a jeden výstup do Simulink. Parametry bude odpor rezistoru a kapacita kondenzátoru.

```
component integracniclanke
    nodes
        vstup= foundation.electrical.electrical;
        vystup= foundation.electrical.electrical;
    end
    outputs
        Vout={ 0, 'V' }; %Odpor:top
    end
    parameters
        Rparam={ 1, 'Ohm' }; %Odpor
        Cparam={ 1e-6, 'F' }; %Kapacita
    end
    components (Hidden=true)
        R = foundation.electrical.electrical.resistor(R=Rparam);
        C = foundation.electrical.electrical.capacitor(C=Cparam);
        Vmetr = foundation.electrical.electrical.voltage;
        %Vytvoření instancí jednotlivých komponent
    end
    connections
        connect(vstup,R.p);
        connect(R.n,C.p,Vmetr.p);
        connect(C.n,vystup,Vmetr.n);
        %Vnitřní zapojení integračního článku s voltmetrem
        connect(Vmetr.V,Vout);
        %Připojení výstupu voltmetru na výstup bloku do simulink
    end
end
```

Je zřejmé, že tato metoda je opravdu jednoduchá a i složení složité součást z již existujících součástí je velice jednoduché.

3.4.3 Směr signálu, polarita portů a referenční bod

Na některé součásti, které chceme vytvořit, může mít vliv směr různých veličin. Jak jsem již zmiňoval toho je dosaženo pomocí funkcí `across` a `through` v rámci MATLAB funkce `setup` (viz. 3.2.4). Obecně ve stejném tvaru pro obě funkce, jako

```
through(PromenaVariables, NodeVstup.OdkazNaPromenouDoFyzikalniOblasti,  
NodeVystup.OdkazNaPromenouDoFyzikalniOblasti);
```

Jednoduchým příkladem může být voltmetr připojený ve stejnosměrném obvodu, po připojení elektrod jednou změříme kladnou hodnotu a po jejich přehození zápornou hodnotu. Je tedy velice důležité správně definovat tyto funkce v kódu `component`. Dále je také možné vytvořit referenční bod (např. zem v elektrických systémech, nádrž v hydraulických, výfuk v pneumatických...), který je značena jako prázdný vektor „[]“.

Připojení elektrického obvodu na zem by mohlo vypadat následovně

```
component zem  
nodes  
    vstup = foundation.electrical.electrical; % :top  
end  
variables  
    ikomponenty = { 0, 'A' };  
end  
function setup  
    through(ikomponenty, vstup.i, []); % Proud ikomponenty je přiřazen proudu portu  
    vstup, který je instancí elektrické fyzikální oblasti  
end  
equations  
    vstup.v == 0;  
end  
end
```

3.4.4 Obrázek, popisky vstupů/výstupů, název bloku komponenty

Tématickou problematiku a názvy bloků jsem již začal v předchozích kapitolách, v této kapitole se na tuto problematiku podíváme podrobněji a přibude k tomu ještě vytvoření obrázku komponenty. Samotný název komponenty je uveden hned za počátkem bloku `component`, v textové poznámce MATLAB. Není vhodné používat názvy s diakritikou, ale je možné použít velkých písmen, mezer a některých přídatných znaků (jako je `+ - * ! @ | # $ % ^ ' : & * () _ . , > < { } []`) pouze s výjimkou / tento znak lze v určitých situacích použít, ale není to vhodné z důvodu, že se tento znak používá při odkazování na jednotlivé bloky v modelu). V případě že není název komponenty uveden název, který bude zobrazen v Simulink bude název souboru v kterém je uložen kód komponenty.

Jméno následující komponenty v Simulink bude Muj Odpor

```
component odpor
    %Muj Odpor
end
```

Jméno následující komponenty v Simulink bude odpor

```
component odpor
end
```

Popis bloku zobrazený po rozkliknutí je umístěn v dalších řádcích za názvem, vytvořením prázdného řádku symbolizujeme nový odstavec. Přerušením komentáře nekomentářovým řádkem popis bloku končí.

```
component odpor
    %Muj Odpor
    %Toto je můj odpor
    %
    %Parametrem je R

    %Toto už není součástí popisu bloku
end
```

Dalším popiskem zobrazeným v Simulink po rozkliknutí bloku, jsou aliasy parametrů, nebo proměnných. Tento alias se zadává za inicializaci parametrů nebo proměnných.

```
parameters
    parametr = { PocHodnota, 'Jednotka' }; %NazevParametru
end
```

Simscape bloky umožňují použití vlastních obrázků ve formátech „.jpg“, „.png“, „.bmp“, jiné formáty obrazu nejsou podporovány. Vhodná velikost obrázku se liší podle počtu portů, ale nejčastěji je volen čtverec 40 na 40 pixelů, nebo obdélník 40 na 60 pixelů a to buď na šířku, nebo na výšku podle komponenty. Výsledná velikost bloku bude odpovídat velikosti obrazu. Samotný soubor s obrazem musí být umístěn ve stejné složce jako soubor s kódem a musí mít stejný název. Například při komponentu pružiny bude zdrojový soubor „pruzina.ssc“ a obrazový soubor bude uložen v „pruzina.bmp“.

Vlastnosti obrázku se mění za názvem v komentáři bloku komponent zobrazovaném v Simulink, jednotlivé parametry jsou odděleny dvojtečkami. Měřítko je určeno první proměnnou měřítko můžeš být jakékoliv celé nebo desetinné číslo s desetinou tečkou (např. 0.5 je polovina velikosti, 2.0 je dvojnásobná velikost). Parametr rotace slouží pro nastavení obrazu na bloku, a sice jestli bude rotovat s blokem, nebo bude ve stejné pozici při rotaci bloku, jsou dvě možnosti **rotates** (obraz rotuje s blokem) nebo **fixed** (obraz nerotuje s blokem).

```
component nazev
    % VlastniNazev : Meritko : Rotace
end
```

Nejdůležitější na každém komponentu jsou asi popisky vstupů a výstupů. Popisky se zapisují do poznámky v blocích kódu **nodes**, **inputs**, **outputs** za jednotlivé instance tříd **domain**, u **inputs** a **outputs** potom za definice počáteční hodnoty a jednotky. Kde první se napíše název, který se zobrazí u vstupu/výstupu a za dvojtečkou následuje jeho pozice. V případě více portů se stejnou pozicí první napsaný v kódu bude první zobrazen v bloku a pod ním/vedle něho bude druhý napsaný. Možné pozice jsou čtyři podle anglických slov top (nahore), bottom (dole), left (vlevo) a right (vpravo).

```
component nazev
    nodes
        %Port vlevo s názvem +
        a = foundation.electrical.electrical; % +:left
        %Port vpravo s názvem -
        b = foundation.electrical.electrical; % -:right
        %Port dole bez názvu
        c = foundation.electrical.electrical; % :bottom
        %Port nahore s názvem hodnota
        c = foundation.electrical.electrical; %hodnota:top
    end
end
```

SEZNAM POUŽITÉ LITERATURY

- [1] THE MATHWORKS, Inc. *MATLAB/Simulink: Simulation and Model-Based Design* [online]. [cit. 2015-01-16]. Dostupné z: <http://www.mathworks.com/products/MATLAB/Simulink/>
- [2] HUŠEK, Petr. *Analogie systémů* [online]. 2013 [cit. 2015-04-16]. Dostupné z: https://moodle.dce.fel.cvut.cz/pluginfile.php/1446/mod_resource/content/2/Pred_3new.pdf
- [3] NOSKIEVIČ, Petr. *Modelování a identifikace systémů*. Ostrava: MONTANEX, 1999. ISBN 80-7225-030-2.
- [4] JOHANSSON, Karl. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. In: *Control Systems Technology*. IEEE, 2000.
- [5] THE MATHWORKS, INC. *Simscape: User's guide* [online]. Sedmnácté vydání. [cit. 2015-05-18]. Dostupné z: http://www.mathworks.com/help/pdf_doc/phymod/simscape/simscape_ug.pdf