Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

# BAKALÁŘSKÁ PRÁCE

Vývoj metod pro pokročilou integraci simulačních nástrojů na bázi standardu „FMI 2.0 for Model Exchange“ a její validace na modelu vybraného energetického zařízení

Plzeň, 2017 Milan Vosáhlo

# P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 19. května 2017

.....

# Poděkování

Tímto bych chtěl poděkovat panu Ing. Martinovi Čechovi, Ph.D. za vedení práce a za věnovaný čas. Dále bych chtěl poděkovat panu Ing. Janovi Reitingerovi za vysvětlení základů práce s OpenModelicou, vstřícnost a pomoc při řešení mnohých technických problémů. V neposlední řadě bych také chtěl poděkovat kolegovi Karlovi Kubíčkovi za spolupráci na projektech a bakalářské práci.

# Anotace

Tato práce vznikla za účelem ověření funkčnosti standardu FMI/FMU, stručného vysvětlení jeho principu, provedení simulací na jednodušším i složitějším modelu a poukázání na problémy, které se vyskytují při práci s tímto standardem při procesu vývoje od Model in the Loop simulací až po Hardware in the Loop. Problémy vzniklé při práci s FMI/FMU standardem i vývojovými prostředími jsou v této práci rozebrány a řešeny.

Jednodušším modelem byl zvolen RLC obvod, složitějším modelem vodní turbína s generátorem. V systému reálného času REX byl vytvořen regulátor pro turbínu a FMU model turbíny i generátoru byl s tímto regulátorem otestován.

Součástí této práce je i návod na vytvoření jednoduchého modelu z fyzikálních bloků v prostředích OpenModelica a MATLAB/Simulink, exportování modelu z OpenModelicy do FMU standardu a porovnání výsledků simulací.

**Klíčová slova:** Functional Mock-up Interface, FMI, FMU, model, přenos modelu mezi prostředími, REX, OpenModelica, SimScape.

# Abstract

The main goal of this thesis is to clarify the functionality of the FMI/FMU standard through the explanation of the basic principles of the Functional Mock-up Interface, the testing of several models and the finding of an explanation - and if possible of a solution - for problems that have occurred while working with this standard in the process of development from the Model in the Loop up till the Hardware in the Loop stages.

Several models of a RLC circuit were made to serve as an instrument for an effective comparison of simulations' results. Also, a water turbine and a generator models were made to test the behaviour of a model representing a more complex system. Finally, the turbine and generator models were used in the REX Control System, to compare the outcome of a real-time simulation with Simulink simulations.

Instructions on how to build a simple model in the OpenModelica and MATLAB/Simulink environments and how to export a model from OpenModelica and convert it to a FMU standard are included in this thesis.



**Keywords:** Functional Mock-up Interface, FMI, FMU, model, transfer of a model from an environment to another, REX, OpenModelica, SimScape.

# Obsah

# Úvod

Functional Mock-up Interface je nezávislý standard umožňující sdílení dynamických modelů napříč různými vývojovými prostředími.

Největší motivací pro vznik tohoto standardu byla potřeba vyřešit situaci, kdy mnoho dodavatelů navrhovalo komponenty v různých vývojových prostředích používajících různé formáty. Vznikem univerzálního standardu se tento problém řeší a jeho používání umožní rychlé ověření, jestli je návrh produktu z komponentů od různých dodavatelů validní. Toto řešení také zlepšuje efektivitu navrhovacího procesu (Software/Model/Hardware-in-the-Loop).

Tato práce se zabývá vytvořením FMU souborů v prostředí OpenModelica, jejich porovnáním s modely v Simulinku a použitím těchto souborů v systému reálného času REX [1].

První část práce stručně popisuje Standard FMI a použitá vývojová prostředí spolu s některými použitými knihovnami a množstvím problémů, které bylo při práci v těchto prostředích potřeba překonat.

Druhá část se zabývá modely RLC obvodu získanými různými přístupy (včetně exportování modelů do formátu .fmu), jejich porovnáním s analytickým řešením rovnice pro tento obvod a sledováním rozdílů v chování modelů při simulacích.

Třetí část slouží jako návod pro vytvoření modelu zmíněného RLC obvodu v prostředích SimScape a OpenModelica. Z OpenModelicy se výsledky následně exportují ve formátu .csv a modely do formátu .fmu a v prostředí MATLAB/Simulink se porovnají.

Ve čtvrté části je využit model vodní turbíny. V prostředí OpenModelica jsou vytvořeny jeho FMU ekvivalenty. Dále byly vytvořeny v Simulinku modely obsahující FMU bloky s turbínou a generátorem a regulátor ze sady RexLib. Tento model byl poté převeden a zpracován v systému REX.

Pokračováním této práce by mohlo být ověření modelu vodní turbíny na reálných datech, vytvoření přesnějšího a složitějšího modelu turbíny a generátoru, vytvoření knihovny fyzických bloků využívajících operační zesilovače v Modelice a Simulinku i jako FMU modely.

Od čtenáře se očekává základní znalost blokového modelování.

# Krátký popis použitých technologií, programů a vývojových prostředí

**FMI/FMU**

Functional Mock-up Interface je nezávislý standard umožňující sdílení dynamických modelů napříč různými nástroji. Mezi významnější firmy, které pro tento standard vyvíjejí knihovny, patří například: Dassault Systemes (Dymola), ESI Group (SimulationX), Modelon (toolboxy pro FMI do Dymoly, MATLABu/Simulinku...) Maplesoft (MapleSim), OpenModelica a dSPACE (v závorkách jsou uvedeny produkty těchto firem). Existují knihovny pro práci s FMI v několika programovacích jazycích, zejména v Pythonu a Javě. Vývoj tohoto standardu byl započat firmou Daimler AG za účelem vylepšení sdílení modelů mezi dodavateli a výrobci. První verze tohoto standardu (FMI 1.0) vznikla v roce 2010, současná verze FMI 2.0 je z roku 2014 a vývoj stále pokračuje. Standard je otevřený s oficiálními hlavičkami a xml schématy na oficiální stránce. Stejně tak jsou otevřené SDK a testovací software, který zjistí, zda je daný FMU soubor v souladu s FMI standardem. [2] [3]

Prakticky jsou FMU soubory ve formátu ZIP se změněnou příponou. Standard má 2 základní módy. Prvním je FMI for Model Exchange. Modely v tomto módu obsahují XML soubor s definicí proměnných a dalšími daty. Funkce v jazyce C představují jednotlivé rovnice v modelu. Druhým módem je FMI for Co-Simulation. Krom výše zmíněného obsahují Co-Simulation modely také solver. FMU funguje na bázi master-slave modelu, kdy některé prostředí ovládá FMU model. [4]

**MATLAB**

Jazyk založený na maticích a numerických výpočtech pro vytváření různých, primárně numerických systémů, analýzu, zobrazení dat a komunikaci s jinými programovacími jazyky. Stejnojmenné vývojové prostředí umožňuje v tomto jazyce pracovat. [5]

**Simulink**

Prostředí v MATLABu pro práci s bloky, umožňující vytváření modelů a simulování dynamických systémů. Obsahuje solvery, knihovnu bloků a komunikuje s MATLABem (načítání proměnných a funkcí a ukládání průběhů simulací).

**SimScape**
Knihovna pro Simulink inspirovaná Modelicou.

**Modelica**
Objektově orientovaný jazyk založený na rovnicích pro komponentové modelování komplexních dynamických systémů. Součástí vývoje tohoto jazyka je i knihovna standardních prvků obsahující přes 1000 fyzikálních bloků a přes 1000 funkcí. [6]

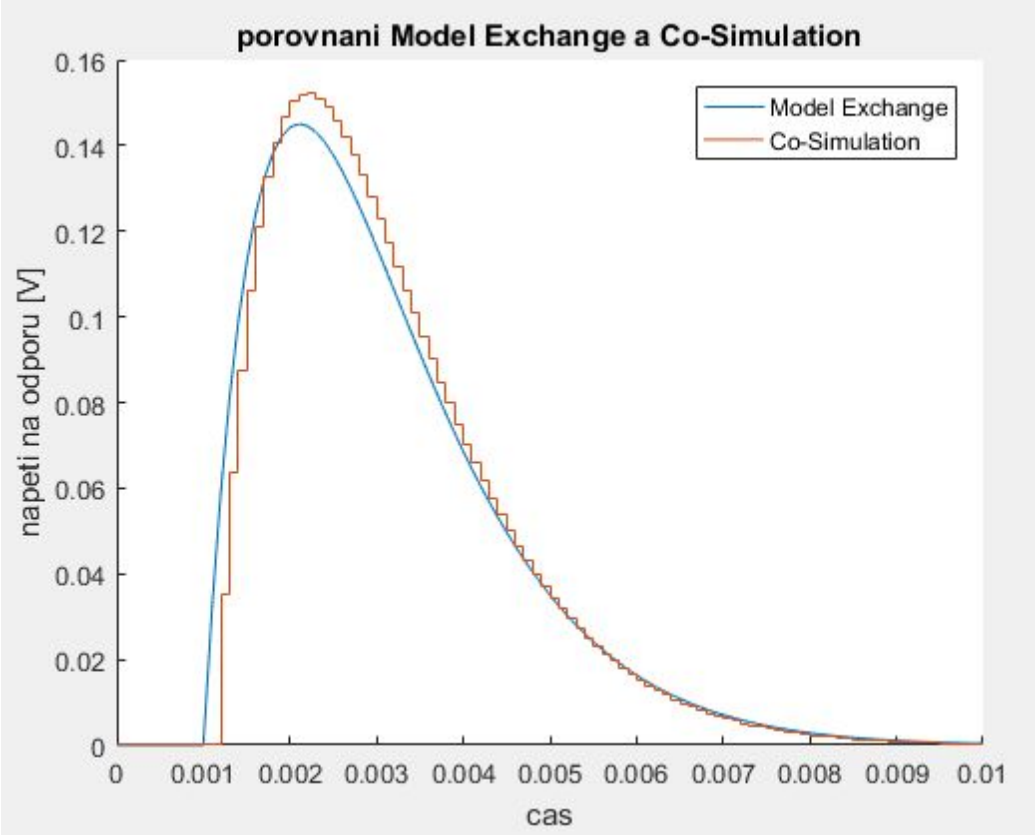
**OpenModelica**
Open-Source prostředí založené na práci s Modelicou pro výzkum, průmysl a akademické použití. [7]

**REX a RexLib**
REX je řídící systém reálného času umožňující simulování modelů. RexLib je knihovna pro Simulink, která obsahuje funkční bloky z prostředí REX. [8]

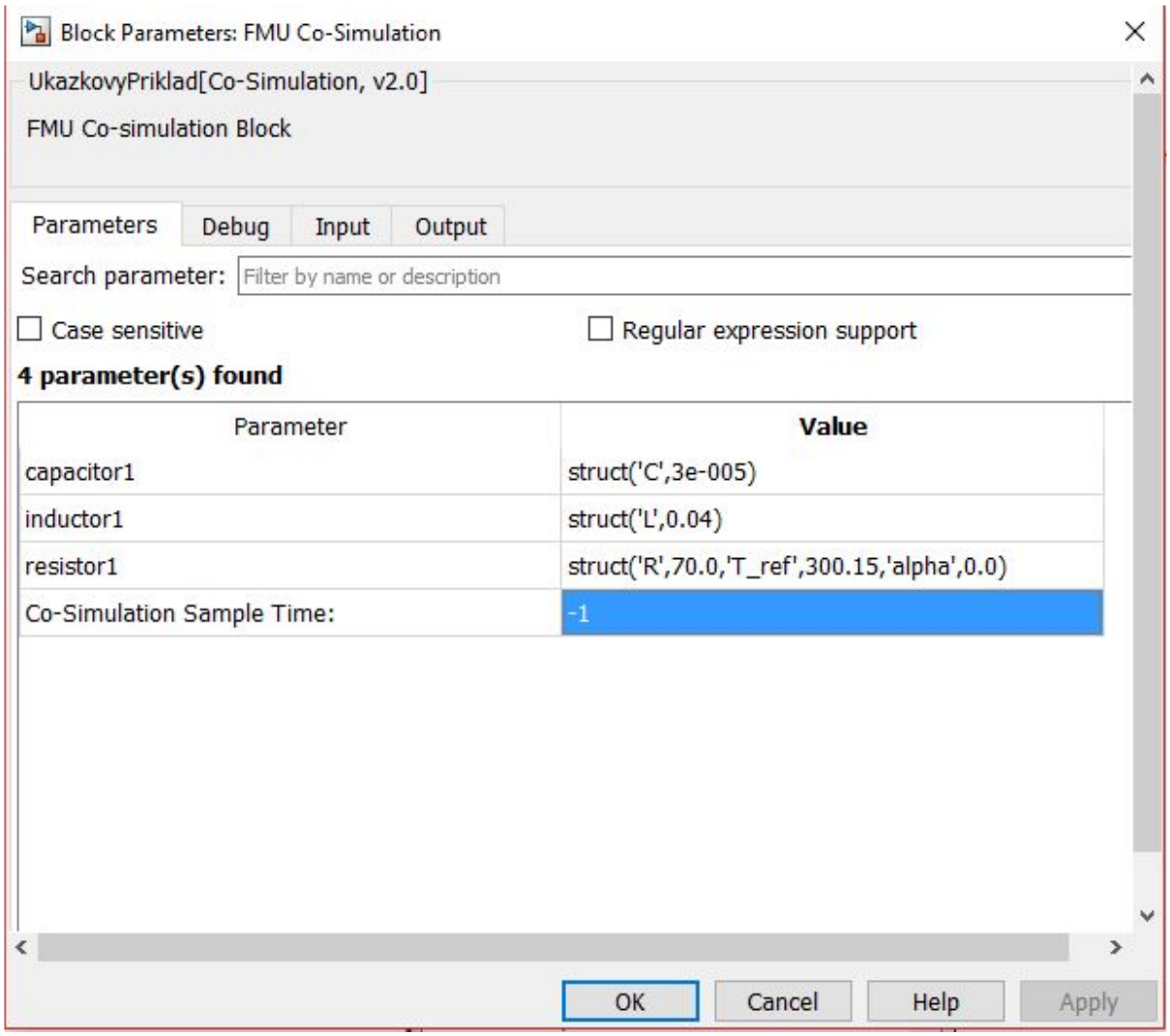
## Problémy objevené během vypracování

**FMU import do MATLABu**
V MATLABu ve verzi 2016 (a starších) není podpora importu FMU. Od firmy MathWorks byla získána betatestovací Pilot Support Package knihovna zahrnující bloky do Simulinku, které mají možnost naimportovat FMU modely. Z počátku byl problém s FMU získanými z OpenModelicy, ten byl ale vyřešen nainstalováním 64bit verze OpenModelicy, oproti 32bit standardní verzi.

**FMU Co-Simulation v Simulinku step time**
Bloky importující Co-Simulation modely nedědí step time od solveru v Simulinku. Po rozkliknutí bloku je ale možné Sample Time změnit. Vykreslené průběhy veličin v simulaci se chovají, jako kdyby obsahovaly zero-order hold (viz obrázek 1), mají tedy schodový průběh, a to nejen při importu FMU z OpenModelicy, ale i při načtení examplů dodaných v Pilot Support Package.




Obrázek 1: Znázornění rozdílu defaultního nastavení FMU Co-Simulation a FMU Model Exchange




Obrázek 2: Možnost nastavení simulační periody v Simulinku

**Derivace vstupu v OpenModelice se nedá exportovat do FMU standardu**

Ačkoliv v prostředí OpenModelica je možné odsimulovat blok, který využívá derivaci vstupu, není možné tento blok exportovat do FMU.

**FMU export z MATLABu**

V současné době nejsou k dispozici knihovny podporující export SimScape modelů do formátu FMU, takže chybí srovnání funkčnosti modelů v FMU s totožnými modely v SimScapu.

**FMU import do Modelicy**

V současné době OpenModelica neumožňuje načíst FMU soubory a pracovat s nimi. Možnost importu v horní liště sice je, nicméně není vůbec funkční. Podle vyjádření

vývojářů se na této funkci pracuje, ale vyřešení všech problémů bude “nějakou dobu” trvat.

### Solvery v MATLABu a OpenModelice

MATLAB a OpenModelica obsahují v základu různé sady solverů. Defaultním solverem v OpenModelice je Dassl, v MATLABu ode45 (Dormand-Prince).

Ostatní solvery v OpenModelice:

- euler
- rungekutta
- dassl
- optimalization
- radau5
- radau3
- impeuler
- trapezoid
- lobatto4
- lobatto6
- symEuler
- symEulerSsc
- heun
- ida
- rengekutta_ssc
- qss

Ostatní Simulink solvery

- diskrétní (bez spojitých stavů)
- ode23 (Bogacki-Shampine)
- ode113 (Adams)
- ode15s (Stiff/NDF)
- ode23s (stiff/Mod. Rosenbrock)
- ode23t(mod stiff./Trapezoidal)
- ode23tb (stiff/TR-BDF2)

Bohužel nelze jednoduše určit, zda mají některé solvery shodnou implementaci, a tak všechny simulace proběhly s defaultními solvery s předpokladem, že vývojáři zvolili

nejlepší dostupný solver. Simulink v některých případech doporučí solver jiný, nicméně se tak nestalo v rámci této práce.

Dalším výrazným rozdílem je například délka kroku v simulačním čase: zatímco v Simulinku je možnost proměnné délky kroku (užitečné v okolí extrémů průběhů), OpenModelica tuto možnost postrádá a má pouze nastavitelnou délku kroku (a to ještě ne zcela spolehlivě).

**Problém OpenModelicy s umístěním souborů**

OpenModelica neumí pracovat s cestami k souborům, které obsahují mezery a některé další znaky. Tento problém se vyskytl v aktuální verzi (1.11.0) a byl pozorován i v několika předchozích verzích.

**Problémové používání OpenModelicy**

Celkově vzato je používání OpenModelicy problematické. Některé operace (například zobrazení třídy prvků) trvají nepřiměřeně dlouho, dochází k pádům programu, část funkcí ještě není implementována, nebo nefungují správně (například některé změny v možnostech nastavení modelu se neprojeví a je třeba změnit přímo kód modelu). Dalším problémem je neintuitivní ovládání (oproti Simulinku například) a zacházení s bloky.

**Používání FMU v REXu**

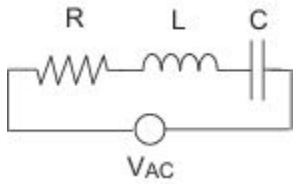
Podobně jako u OpenModelicy je potřeba 64bit verze systému REX pro práci s FMU Co-Simulation soubory.

**Velikost modelů v různých formátech**

Model s fyzikálními komponenty v Simulinku má velikost okolo 20 kB, model stejného systému v Modelice má 3 kB a FMU soubor (Model Exchange i Co-Simulation) exportovaný z OpenModelicy má 11.5 MB.

# RLC obvod

Sériový RLC obvod je jednoduchý elektrický obvod obsahující 4 komponenty: zdroj napětí, rezistor, cívku (inductor) a kondenzátor (capacitor) (viz obrázek 3). Odpor spojovacích vodičů je buď zanedbán, nebo započítán do odporu rezistoru.




Obázek 3: Schéma RLC obvodu [9]

## Získání vztahů pro RLC obvod

Z Kirchhoffových zákonů byl získán vztah:

$$V_{ac} = V_R + V_L + V_C$$

kde $V_{ac}$ je vstupní napětí, $V_R$ je napětí na rezistoru, $V_L$ je napětí na cívce a $V_C$ je napětí na kondenzátoru. [10]

S pomocí Ohmových zákonů byly rozvinuty vztahy pro $V_R$, $V_L$ a $V_C$:

$$V_{ac} = Ri(t) + L\frac{di}{dt} + \frac{1}{C}\int_{-\infty}^{t} i(t)\,dt$$

kde $i(t)$ je proud v obvodu, $R$ je odpor rezistoru, $L$ je indukčnost cívky a $C$ je kapacita kondenzátoru.

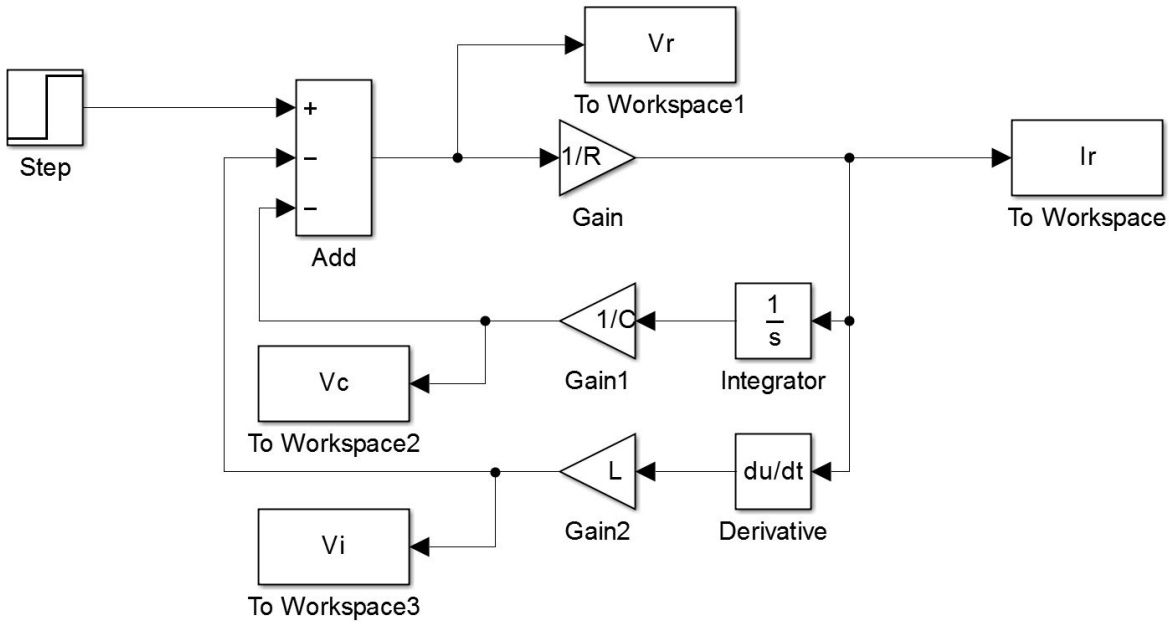
Pro všechny simulace RLC obvodu byly použity následující konstanty:
vstupní napětí $U$ = 0.2 V
odpor rezistoru $R$ = 70 Ω
indukčnost cívky $L$ = 0.04 H
kapacita kondenzátoru $C$ = 0.00003 F
při střídavém proudu frekvence $f$ = 10 Hz

## Diferenciální rovnice v Simulinku

Prvním pokusem o vyřešení diferenciální rovnice bylo vyjádření napětí na odporu:

$$Ri(t) = V_{ac} - L\frac{di}{dt} - \frac{1}{C}\int_{-\infty}^{t} i(t)\,dt$$

Byl vytvořen model v Simulinku pro vyřešení této rovnice (viz obrázek 4):

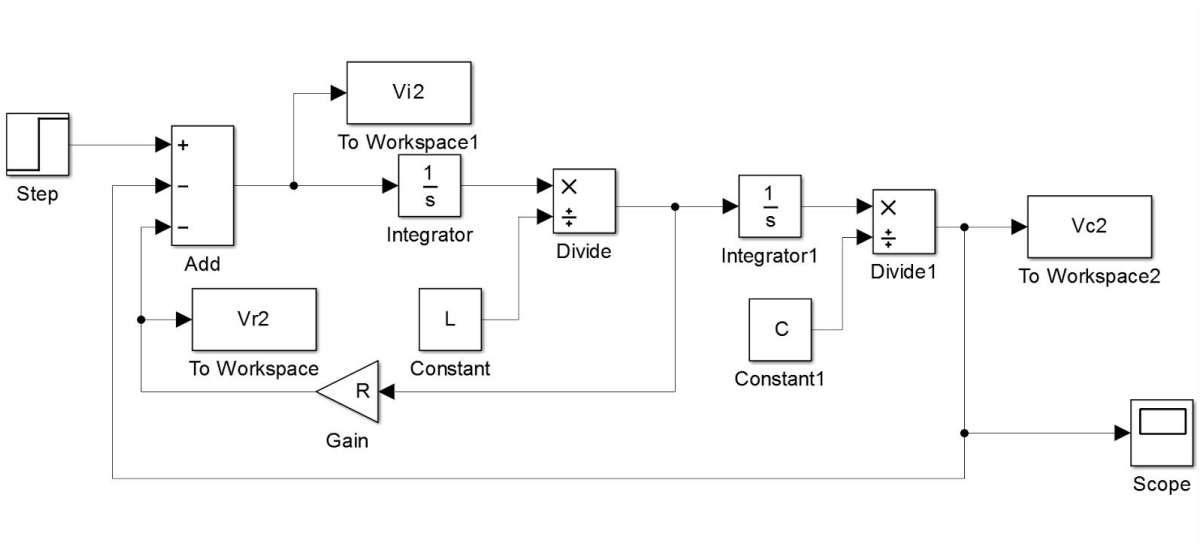



Obrázek 4: Schéma pro řešení implicitní diferenciální rovnice

Kvůli použití derivačního bloku ale vznikla výrazná nepřesnost oproti očekávanému výstupu. Proto byla vyjádřena nejnižší derivace:

$$L\frac{di}{dt} = V_{ac} - Ri(t) - \frac{1}{C}\int_{-\infty}^{t} i(t)\,dt$$

Byl vytvořen model pro tuto rovnici (viz obrázek 5):




Obrázek 5: Schéma pro řešení diferenciální rovnice RLC obvodu

Výstupy tohoto modelu už odpovídaly výstupům ze SimScapu, OpenModelicy i analytickému řešení.

## Diferenciální rovnice v OpenModelice

V OpenModelice byl vytvořen textový model diferenciální rovnice s jedním integračním blokem (viz obrázek 6). [11]

```
16    Vac = L * di + R * i + integralI / C;
17    der(i) = di;
18    Vi = L * di;
19    Vr = R * i;
20    der(Vc) = i / C;
```

Obrázek 6: Zápis diferenciální rovnice pro RLC obvod v jazyce Modelica

## Analytické řešení diferenciální rovnice

Byla vyřešena diferenciální rovnice pro vstup $U = V \cdot sin(f \cdot t)$, kde $V$ je maximální napětí a $f$ je frekvence. [10]

Vztahy pro napětí na resistoru, cívce a kondenzátoru:

$$V_r = R \cdot A \cdot sin(f \cdot t - \varphi)$$

$$V_i = L \cdot A \cdot f \cdot cos(f \cdot t - \varphi)$$

$$V_c = \frac{-A}{f \cdot C} cos(f \cdot t - \varphi)$$

*A* je pomocná proměnná pro výpočet amplitudy a φ je fázový posun. Vztahy pro tyto veličiny:

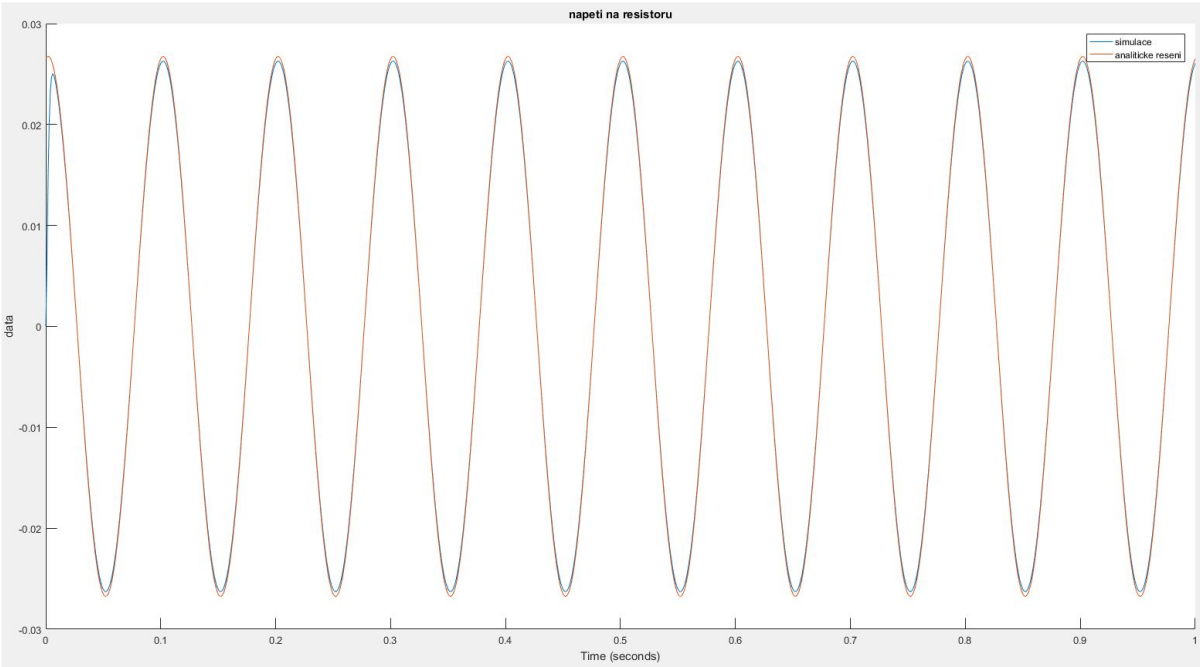
$$A = \frac{f \cdot V}{\sqrt{(L \cdot f^2 - \frac{1}{C})^2 - (R^2 f^2)}}$$

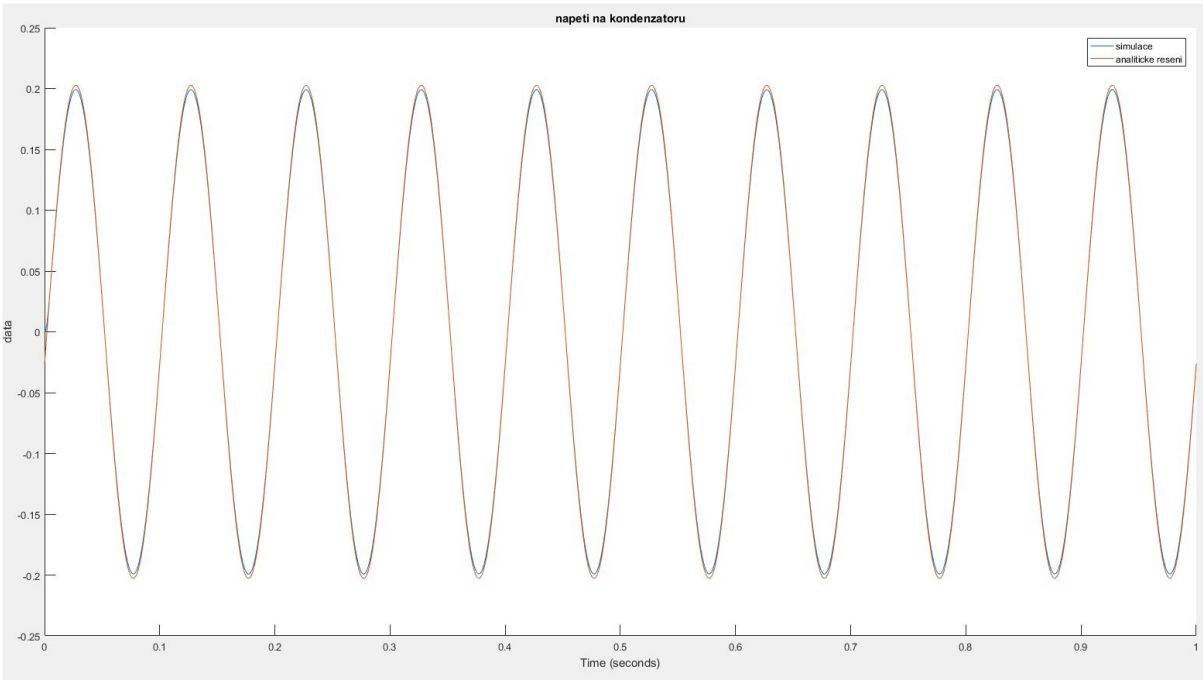
$$\varphi = tan^{-1}(\frac{L \cdot f}{R} - \frac{1}{R \cdot C \cdot f})$$

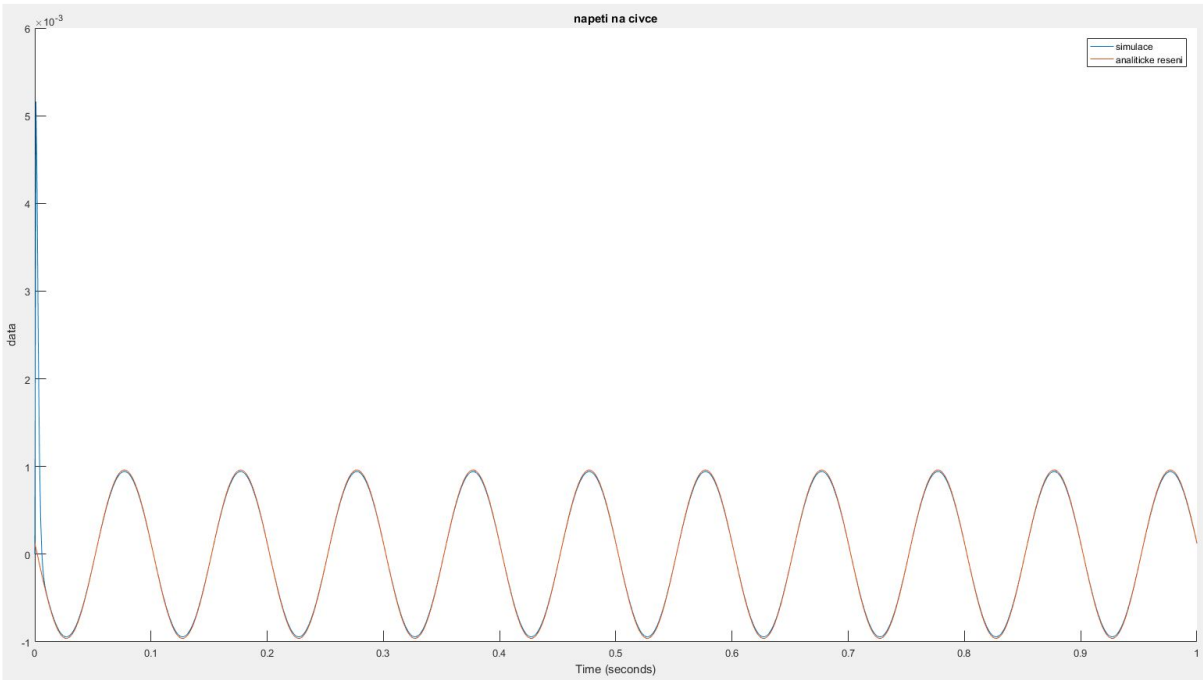
Výpočet v MATLABu:

```
A=(f*V)/sqrt((L*f*f-(1/C))*(L*f*f-(1/C))-(R*R*f*f));
fi=atan((L*f/R)-(1/(R*C*f)));
VrA=R*A*sin(f*t-fi);
ViA=L*A*f*cos(f*t-fi);
VcA=(-A/(f*C))*cos(f*t-fi);
```

Výsledky výpočtů neodpovídají přesně simulacím (amplituda je odlišná, fázový posun je shodný), protože v řešení není zahrnutý přechodový děj (viz obrázky 7-9).






Obrázky 7-9: Průběh napětí na jednotlivých simulovaných komponentech

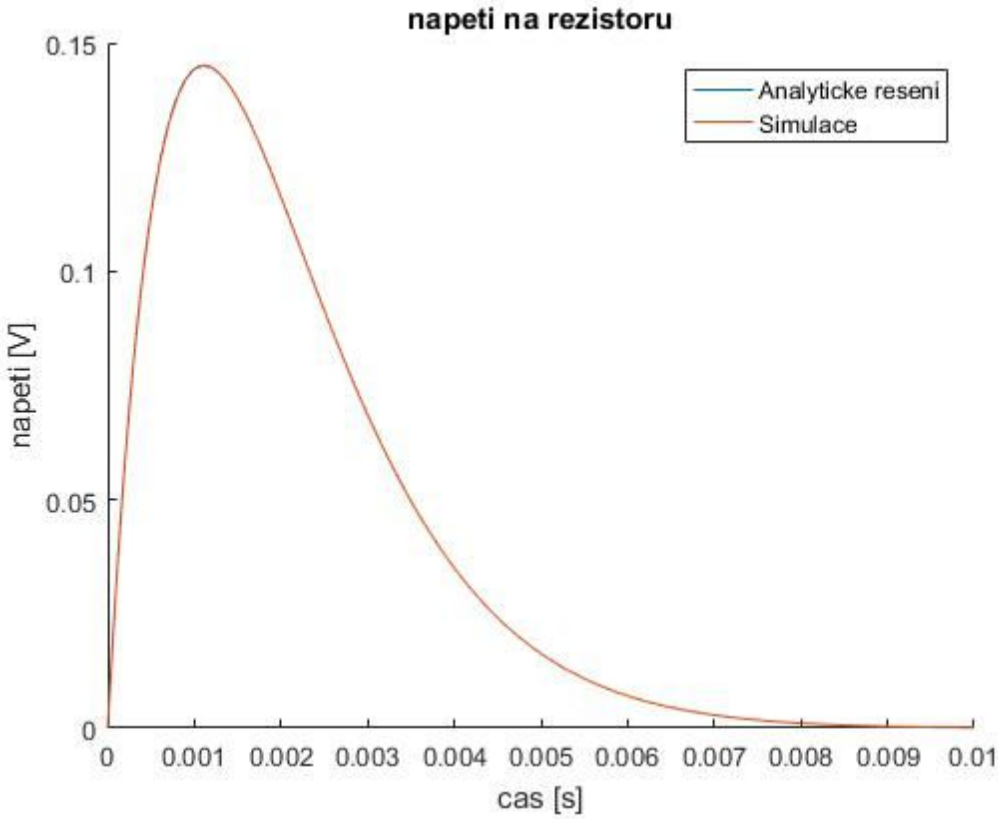
Pro skokový vstup byla rovnice také vyřešena, a to pro skok z nulového napětí na 0.2 Voltu v čase 0.

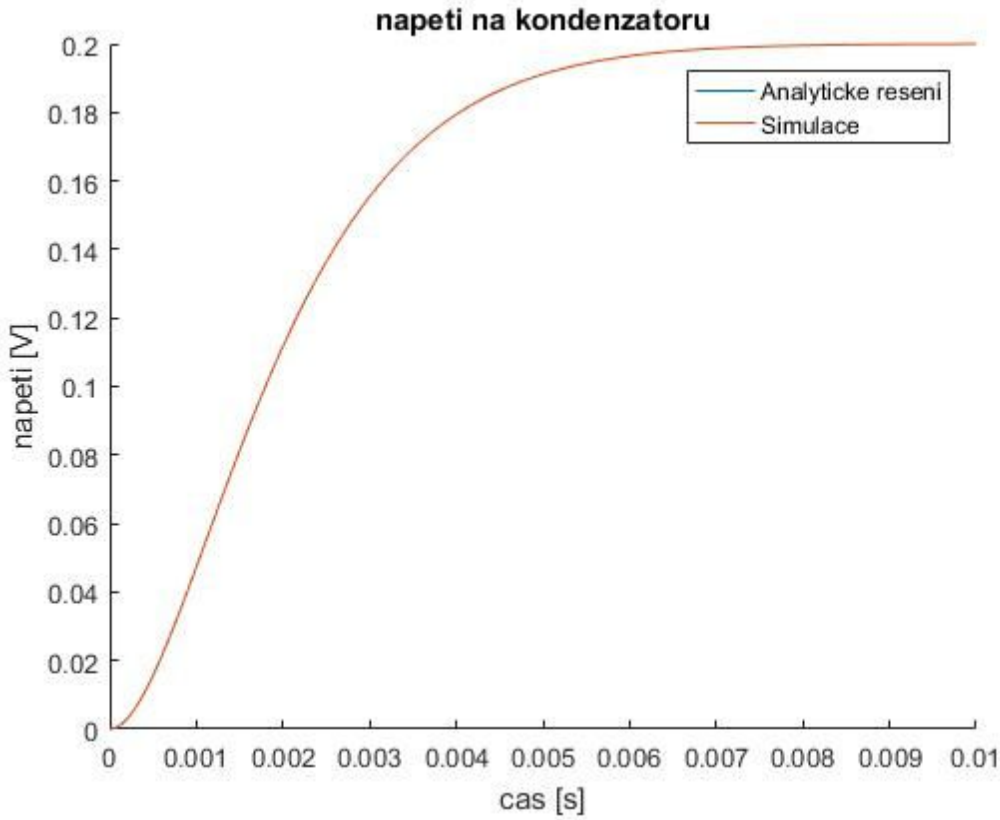
Zápis řešení v MATLABu:

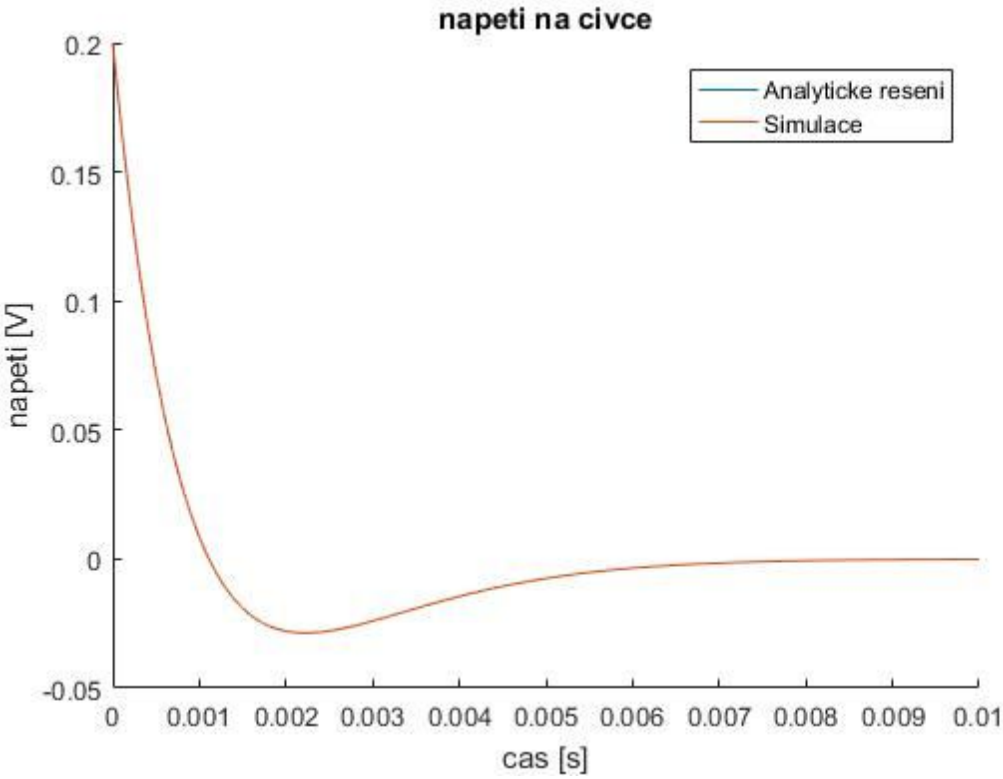
```
I=(3^(1/2).*13^(1/2)*sin((125*3^(1/2)*13^(1/2)*t)/3).*exp(-875
*t))/325;
VrA=R*I;
```

```
ViAp=5*exp(-875*t).*cos((125*3^(1/2)*13^(1/2)*t)/3) -
(35*3^(1/2)*13^(1/2).*sin((125*3^(1/2)*13^(1/2)*t)/3).*exp(-87
5*t))/13;
ViA=L*ViAp;
VcAp=-(3*39^(1/2).*exp(-875*t).*(875.*sin((125*39^(1/2)*t)/3)
+ (125*39^(1/2).*cos((125*39^(1/2)*t)/3))/3))/812500000;
VcA=VcAp/C + V;
```

Porovnání výsledků simulace v Simulinku oproti této rovnici (obrázky 10-12):





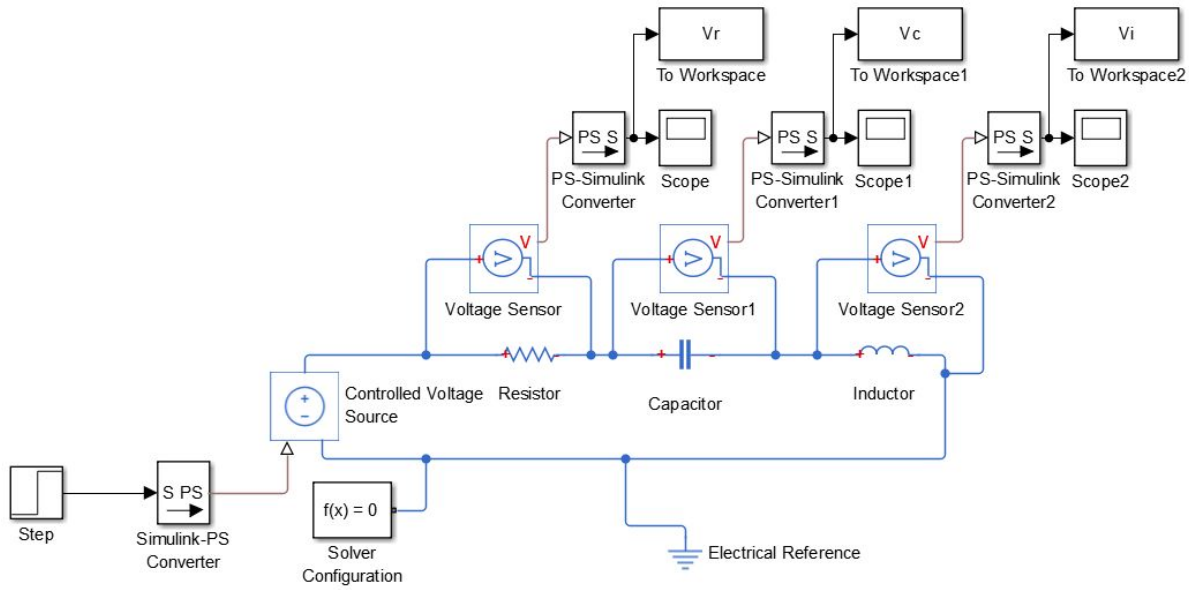

Obrázky 10-12: Porovnání analytického a numerického řešení

Průměrná hodnota odchylky při tomto experimentu s různou délkou kroku byla:

| Délka kroku | 0.0001 | 0.00001 | 0.000001 |
|---|---|---|---|
| Relativní odchylka | 1.3950e-04 | 1.2963e-07 | 1.2826e-10 |

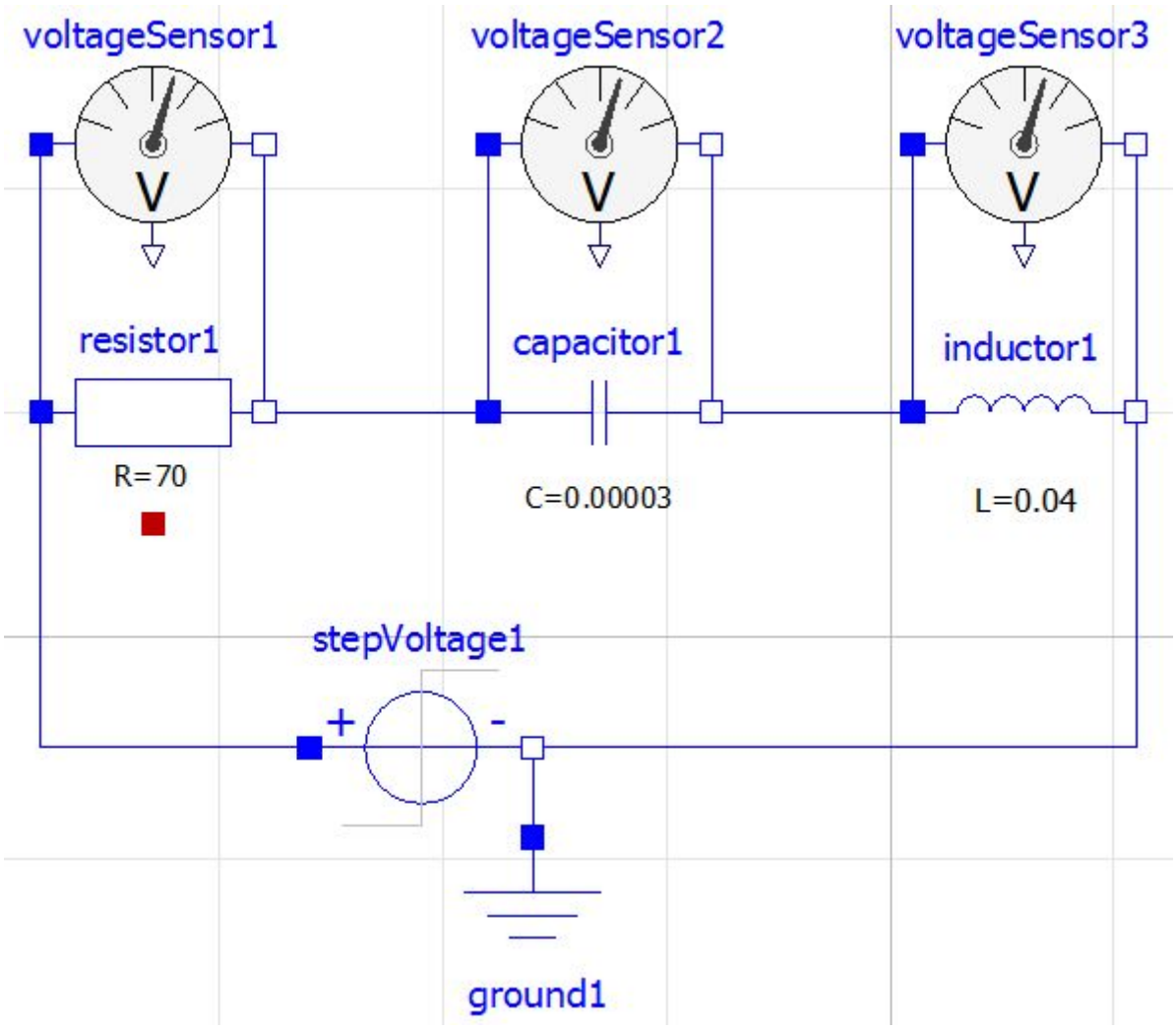
## Model obvodu s užitím SimScape a Modelica bloků

Pro tento obvod byl vytvořen i model obsahující jednotlivé fyzikální komponenty v prostředí Simulink/SimScape (viz obrázky 13 a 14).




Obrázek 13: Model RLC obvodu v prostředí Simulink složený ze SimScape komponentů

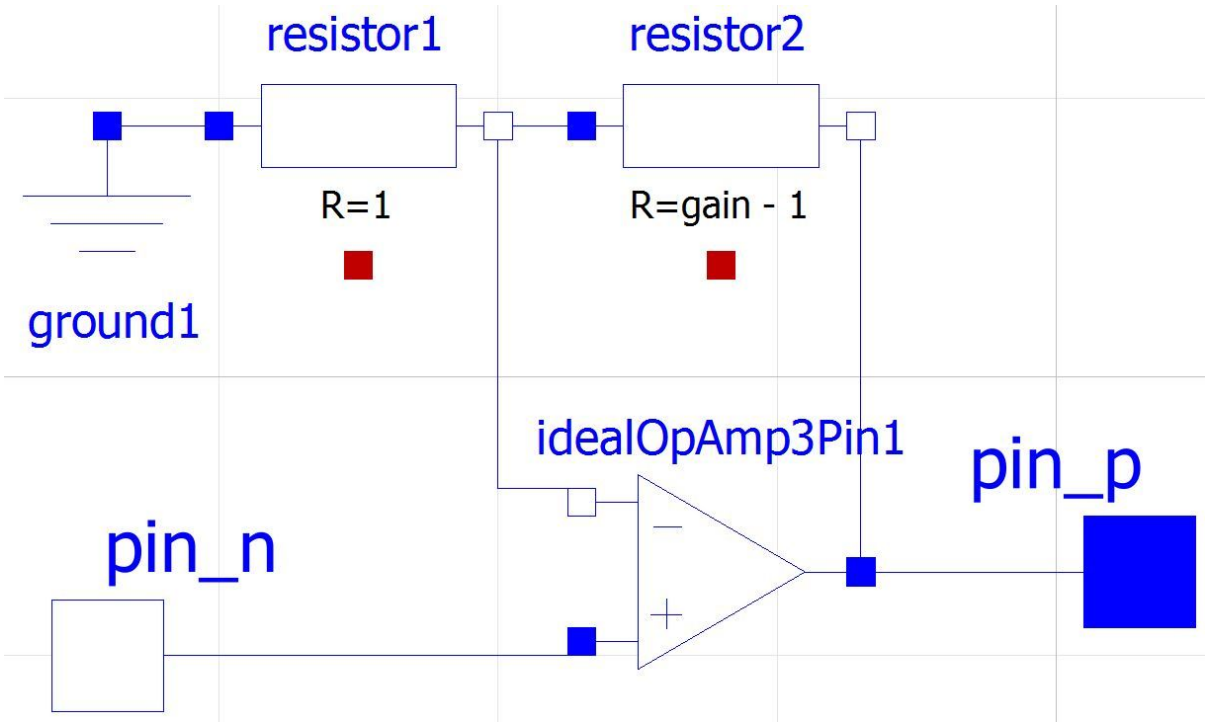
Prakticky stejný model byl vytvořen v prostředí OpenModelica.



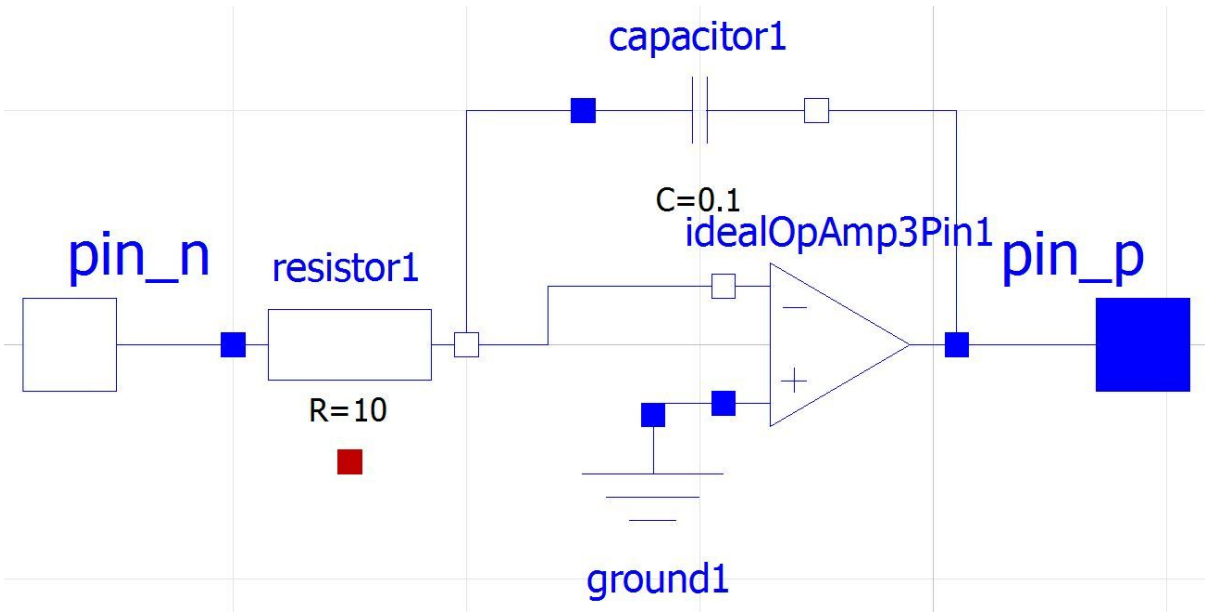

Obrázek 14: Model RLC obvodu v prostředí OpenModelica složený z bloků jazyka Modelica

## Řešení rovnice s užitím operačních zesilovačů

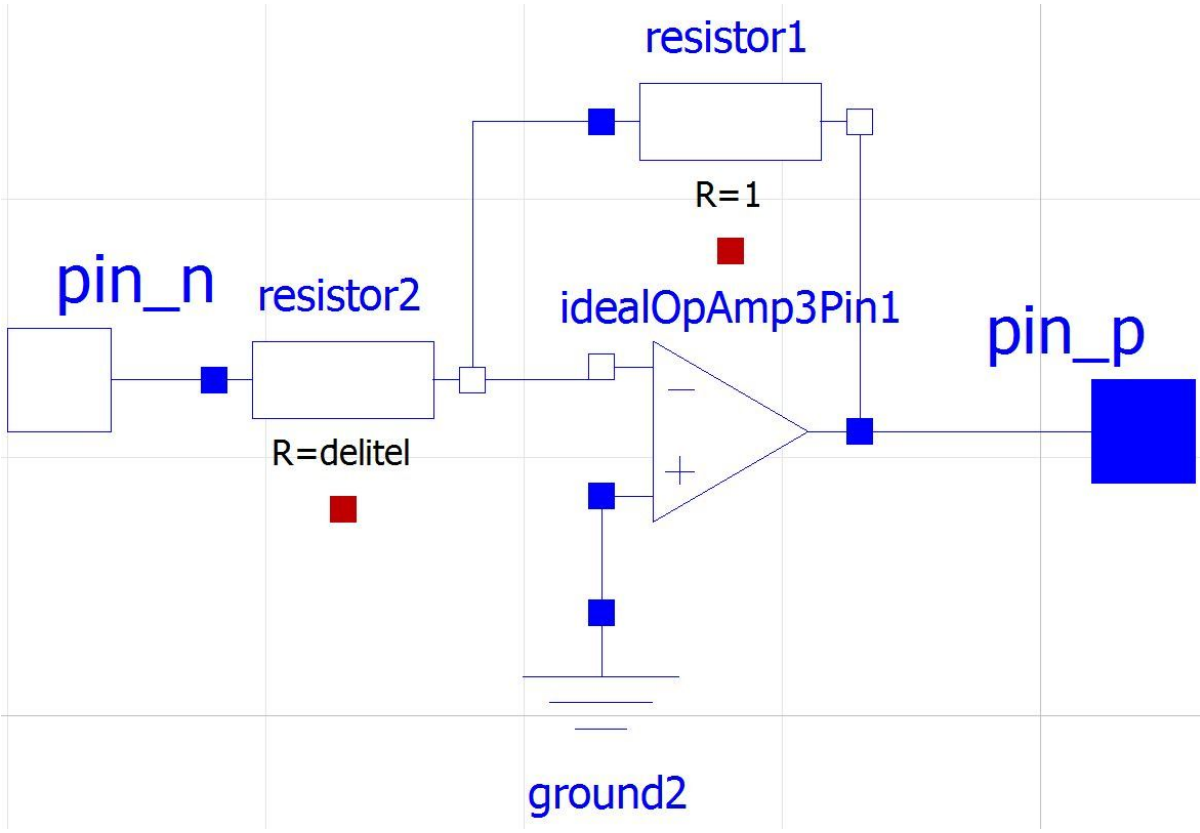
V SimScapu i OpenModelice byly vytvořeny bloky s užitím operačních zesilovačů, které nahrazují některé komponenty Simulinku (viz obrázky 15-18). [12]



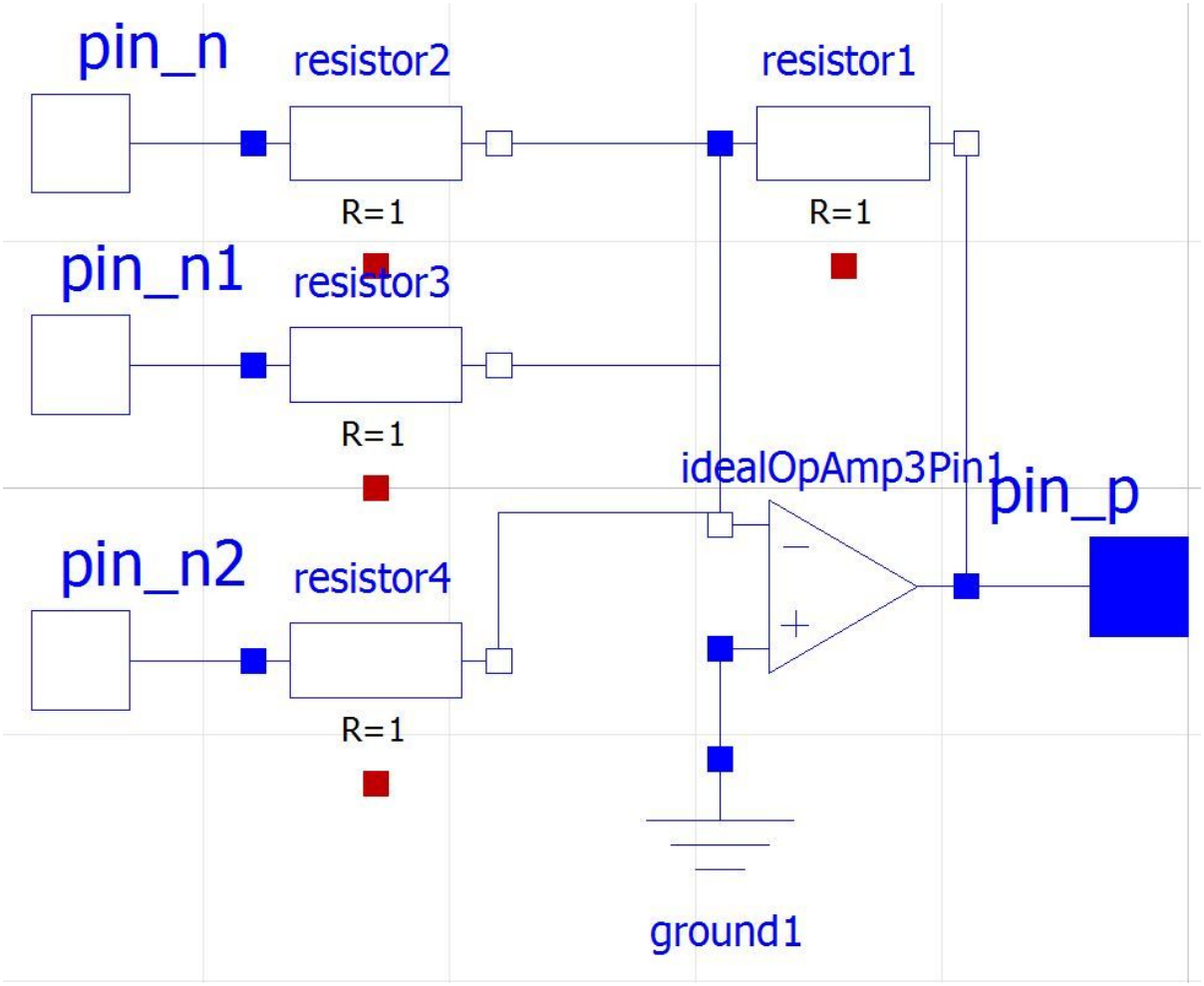

Obrázek 15: Gain - neinvertující zesilovač




Obrázek 16: Integrator - integrační zesilovač

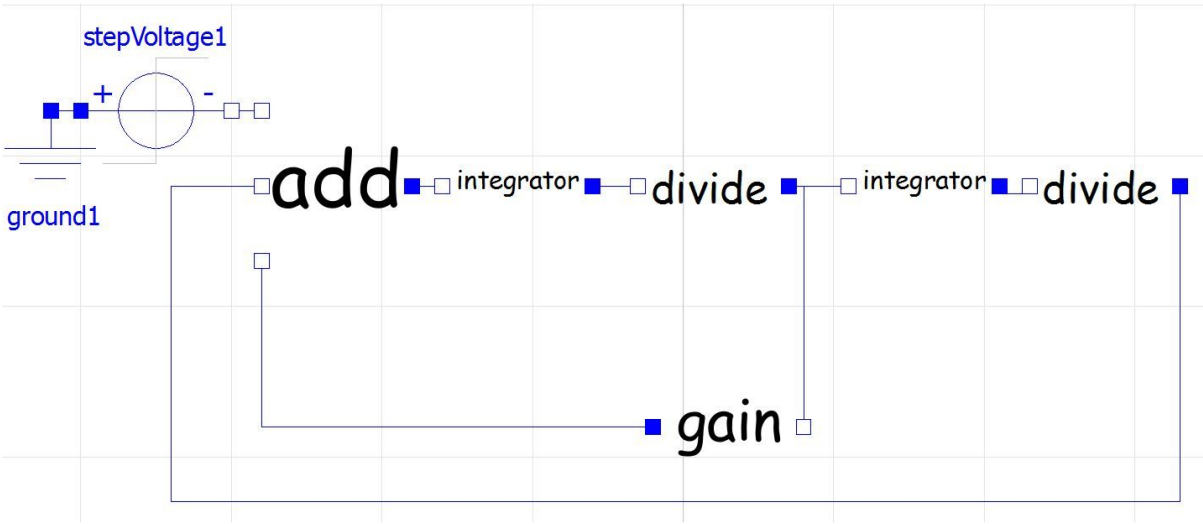



Obrázek 17: Divide - invertující zesilovač

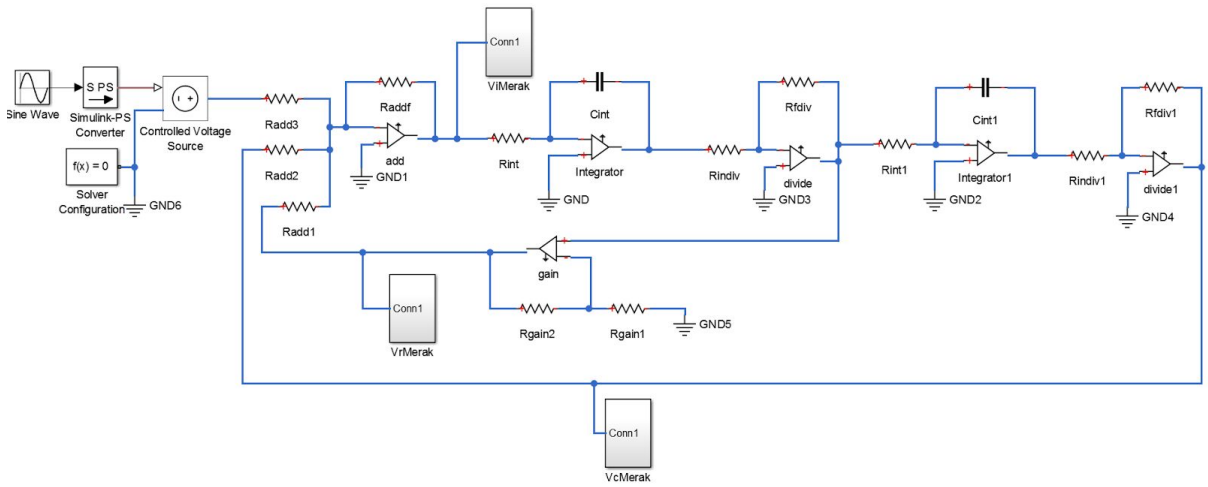



Obrázek 18: Add - sčítací zesilovač

S užitím těchto bloků bylo vytvořeno stejné schéma diferenciální rovnice RLC obvodu jako v Simulinku (viz obrázek 19). Stejný model s operačními zesilovači byl vytvořen i v Simulinku (viz obrázek 20).



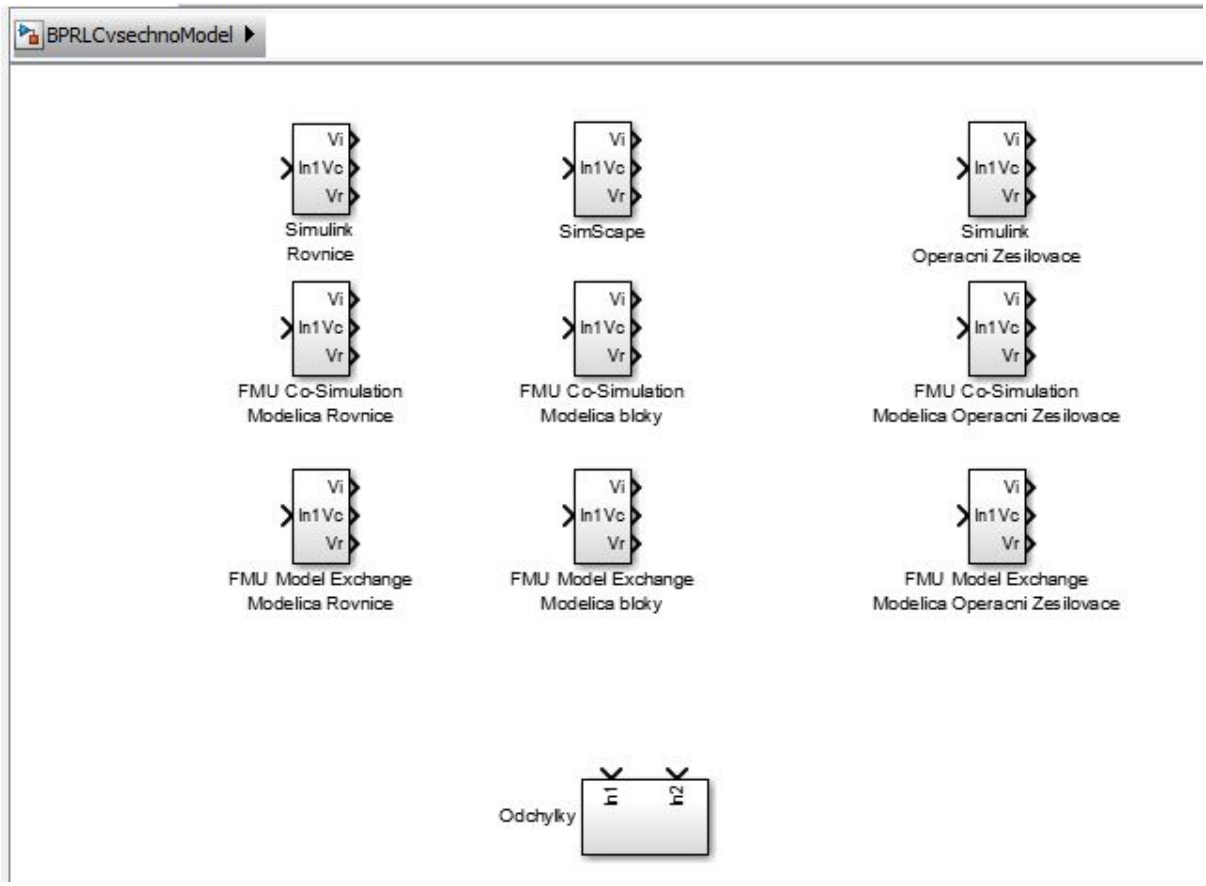

Obrázek 19: Zapojení řešící diferenciální rovnici složené z bloků obsahující operační zesilovače




Obrázek 20: Zapojení s operačními zesilovači v Simulinku

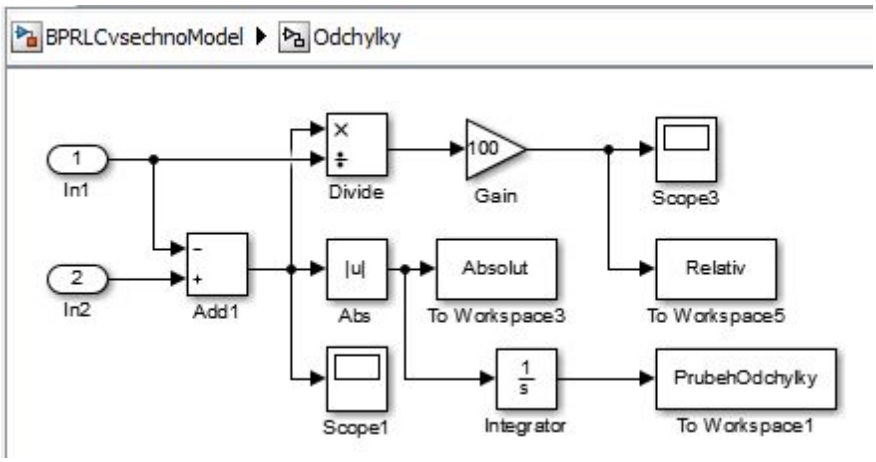
Ze všech modelů v Modelice byly vytvořeny FMU soubory (Model Exchange i Co-Simulation) a byly v Simulinku odsimulovány pomocí Pilot Support Package pro FMI/FMU. Srovnání výsledků je v části níže.

Všechny modely byly jako subsystémy vloženy do jednoho modelu (obrázek 21). K tomu jsou v jednom skriptu pro porovnání načteny i výsledky simulací z OpenModelicy ve formátu .csv.




Obrázek 21: Všechny zmíněné modely jako subsystémy v prostředí Simulink

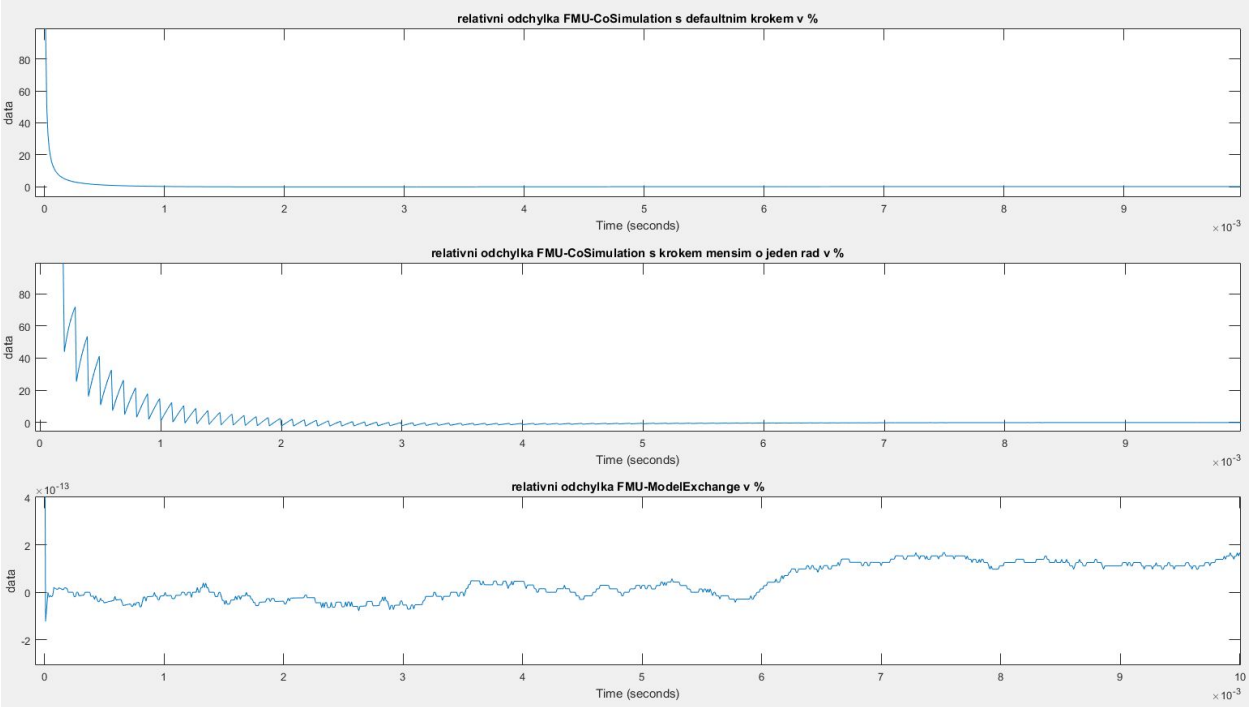
Blok určující odchylky (viz obrázek 22):




Obrázek 22: Schéma výpočtů odchylek dvou vstupních signálů

Pro všechny tyto modely byla odsimulována odezva na krokový skok z nuly na napětí 0.2 V. Při tomto experimentu byl nastavený pevný krok 0.00001 s. Jako referenční model byl zvolen model simulující diferenciální rovnici v Simulinku a všechny uvedené odchylky jsou relativní odchylky v procentech vůči tomuto modelu.

Porovnávanou veličinou bylo napětí na kondenzátoru, protože průběh této veličiny je vzdálen od nuly, tudíž nedochází k numerickým nepřesnostem, které by vznikly při dělení číslem blízkém nule. Protože mají všechny modely podobný průběh odchylky, jsou níže vypsány průměrné hodnoty těchto odchylek, jakožto nejvíce vypovídající hodnoty. Pro FMU Co-Simulation modely obsahuje tabulka pro každý model více hodnot pro různé délky kroků v nastavení těchto bloků. Byly použity tři nastavení: defaultní (-1), o řád větší krok a krok shodný s krokem simulace.




Obrázek 23: Příklad průběhu relativních odchylek pro různé modely. Na počátku při skoku dochází k přechodovému ději a odchylka nabývá velkých hodnot, poté se modely ustálí a odchylka se přiblíží nule

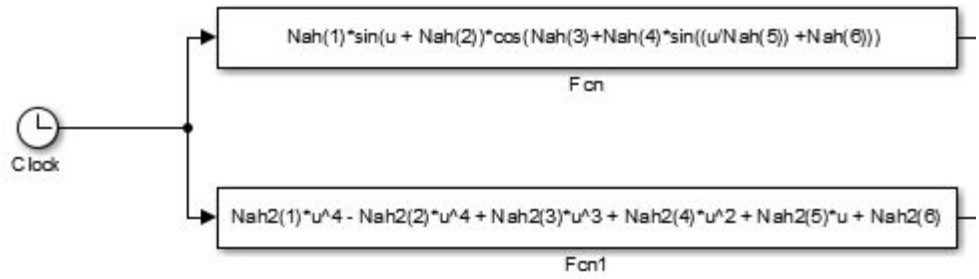
Odchylky pro modely řešené simulinkovským solverem.

| Model-Exchange rovnice | SimScape bloky | Model-Exchange bloky | SimScape Operační zesilovače | Model-Exchange Operační zesilovače |
|---|---|---|---|---|
| 3.0948e-14 | 6.0729e-14 | 3.7589e-14 | 8.0123e-15 | 2.9161e-15 |

Tabulka s Co-Simulation modely:

| Délka kroku bloku | Rovnice | Fyzikální komponenty | Operační zesilovače |
|---|---|---|---|
| Default | 0.1231 | 0.2645 | 1.2846 |
| 0.0001 | 3.7880 | 4.5529 | 7.8352 |
| 0.00001 | 0.2508 | 0.3624 | 1.2859 |

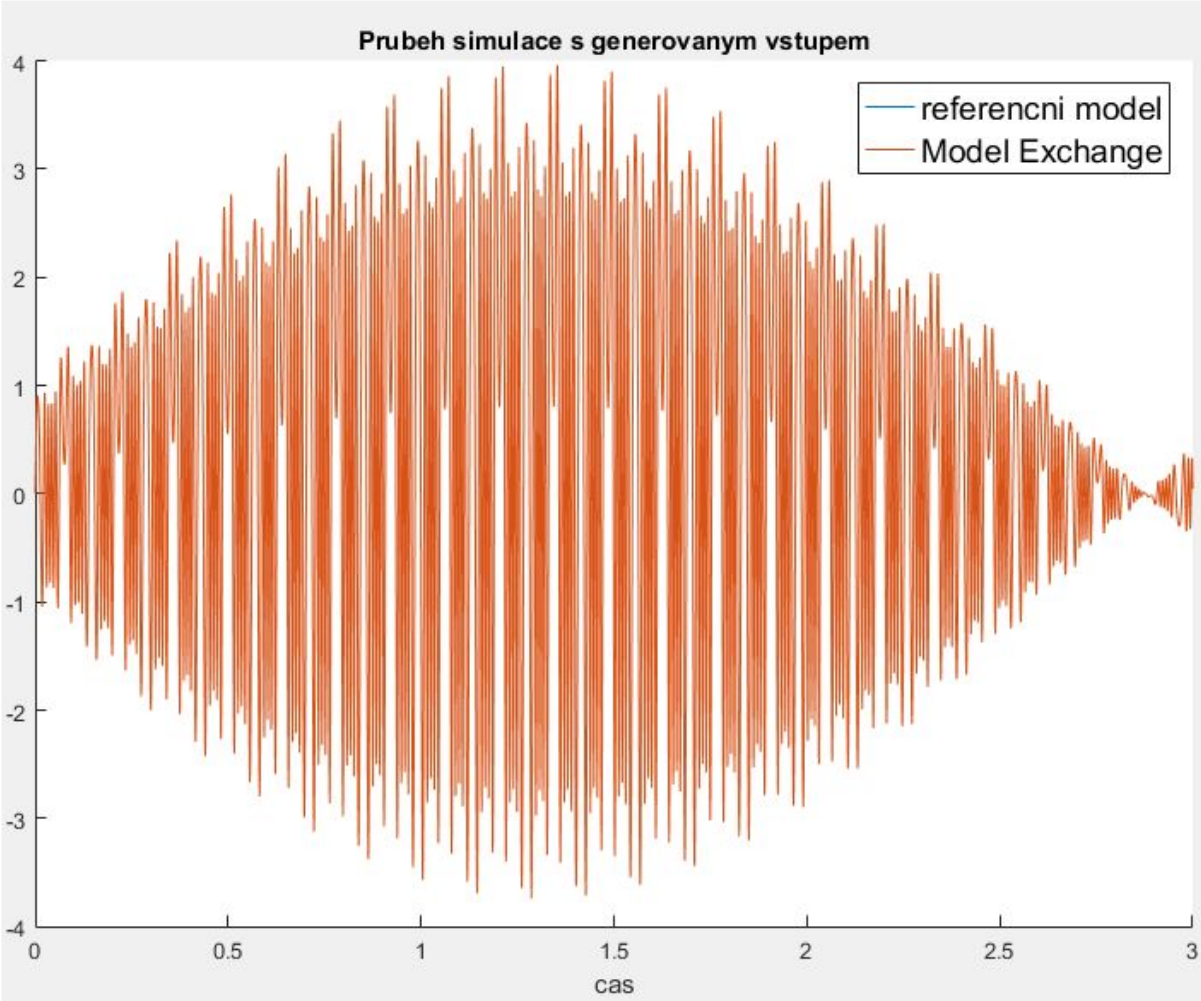
Pro možnosti dalšího zkoumání bylo vytvořeno několik generátorů signálu.
Základní vytvořené generátory náhodného signálu (viz obrázek 24):




Obrázek 24: Generátory náhodných spojitých vstupů

Náhodné proměnné jsou v MATLABu získány následovně:
Nah= abs(10.*randn(6,1)+1);
Nah2=5.*randn(6,1);

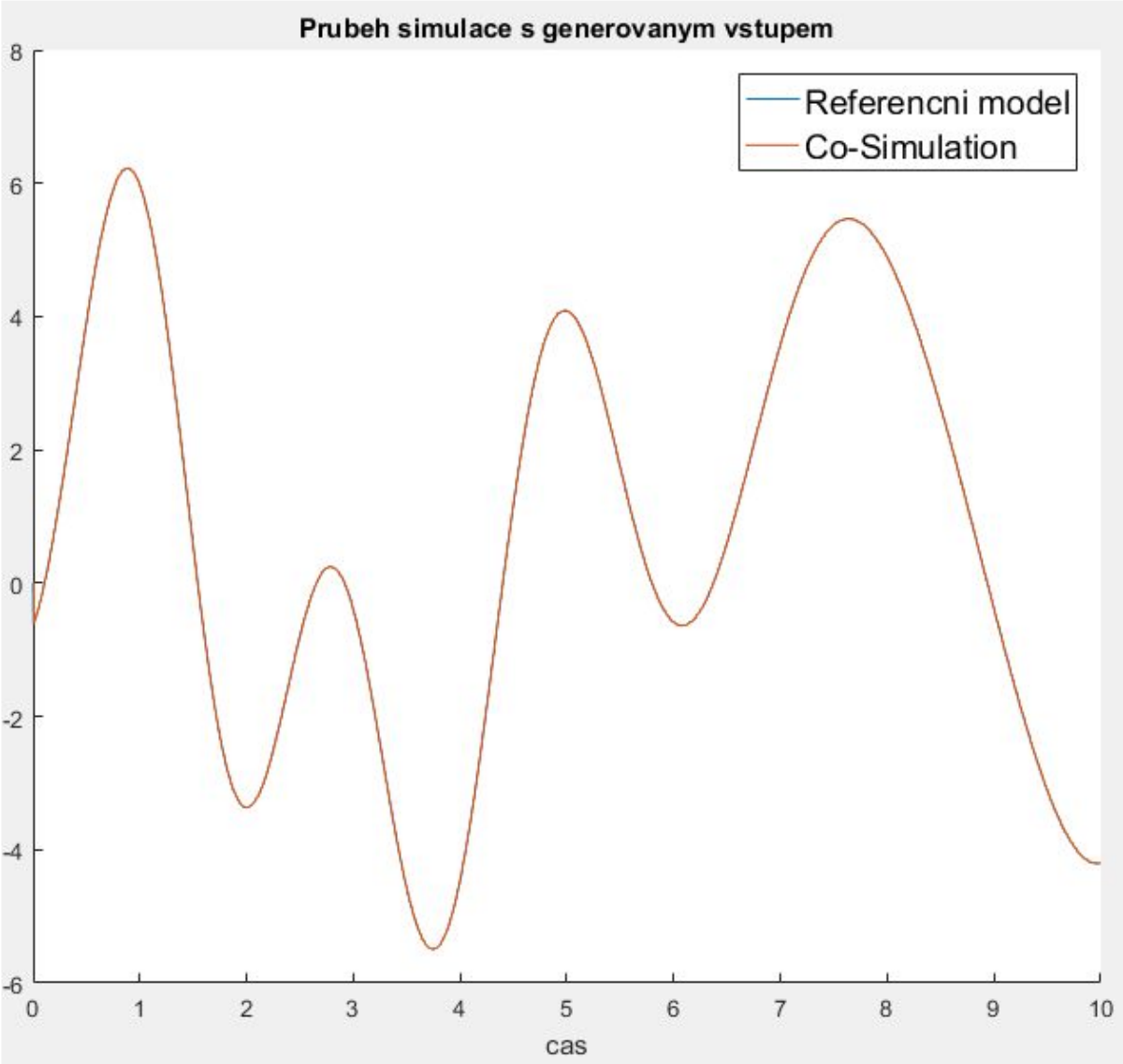
Průběh simulace s tímto generátorem je pak vidět na obrázcích 25 a 26. S různými vygenerovanými hodnotami jsou výstupy tohoto generátoru různé křivky s poměrně klidnými průběhy, ale také mohou být podobné šumům. Opakované měření odchylek s různými instancemi tohoto generátoru může sloužit k validaci chování modelů s různorodými vstupy.




Obrázek 25: Průběh sledovaných veličin s náhodným generátorem vstupu

Jak je vidět, odchylka mezi modelem v Simulinku a FMU Model Exchange modelem není zřetelná. Průměrná hodnota velikosti relativní odchylky při tomto experimentu byla 6.4627e-11. Relativní odchylka se sledovatelně projevila pouze v případě, kdy se sledované veličiny přiblížili k nulové hodnotě.

Pro další experiment byly vygenerovány jiné náhodné hodnoty (viz obrázek 26). Tentokrát byl druhý model FMU Co-Simulation a průměrná hodnota velikosti relativní odchylky při tomto experimentu byla 5.6690e-04.




Obrázek 26: Průběh sledovaných veličin s náhodným generátorem vstupu

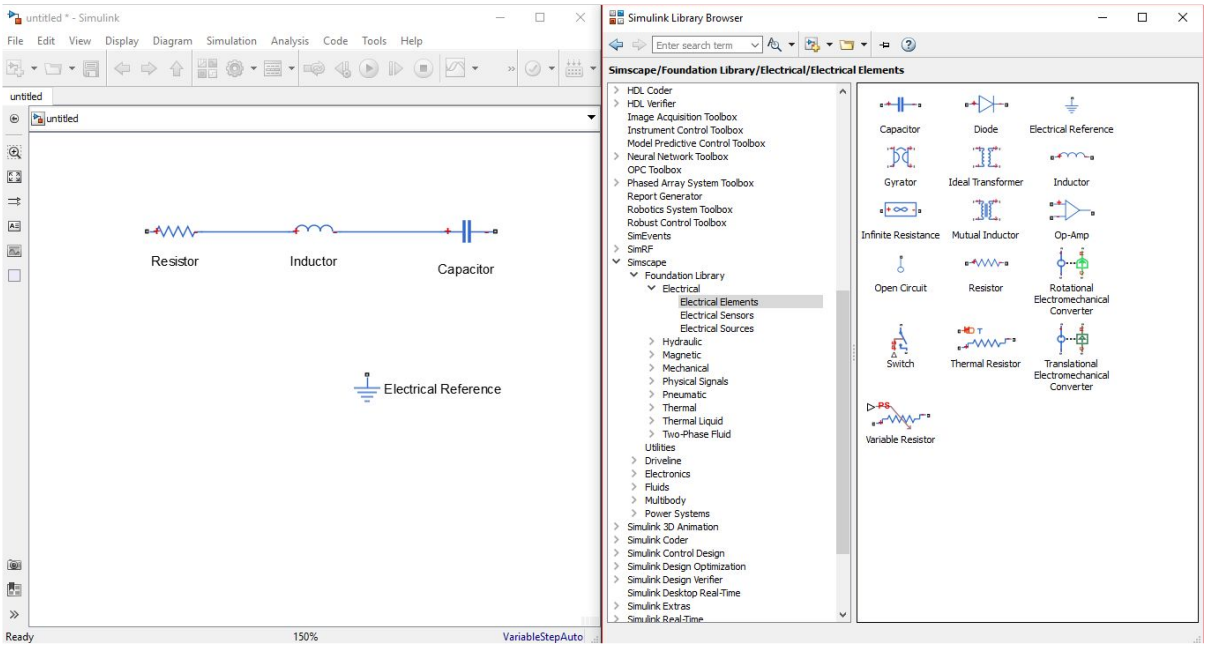
# Návod na sestavení RLC obvodu

Tato část práce se zaměřuje na návod na odsimulování jednoduchého modelu v Simulinku a OpenModelice a porovnání jejich výsledků.

## Postup pro Simulink

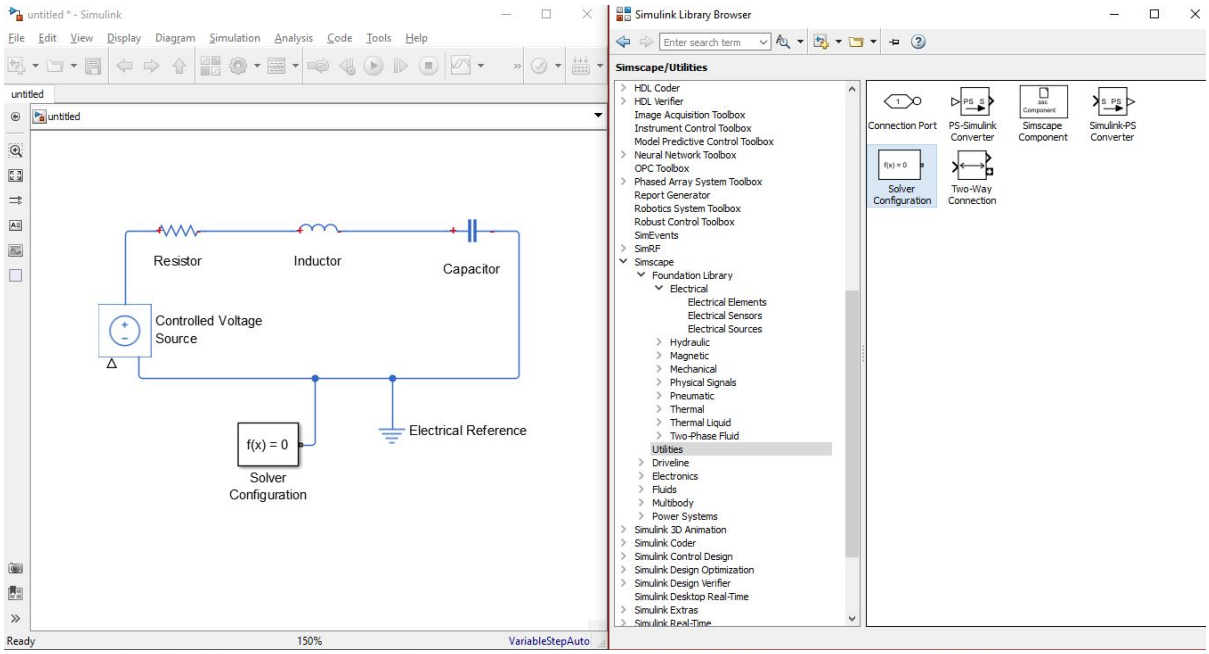
Začneme s hlavními prvky obvodu, budeme pokračovat přidáním měřících komponentů a nakonec dodáme simulinkovské komponenty pro ovládání zdroje a uložení průběhů sledovaných veličin.
Nejprve v Simulinku přidáme do modelu z knihovny *SimScape → Foundation Library → Electrical → Electrical Elements* bloky *Capacitor, Inductor, Resistor* a *Electrical Reference*.

První tři zmíněné bloky zapojíme sériově jako na obrázku 27:



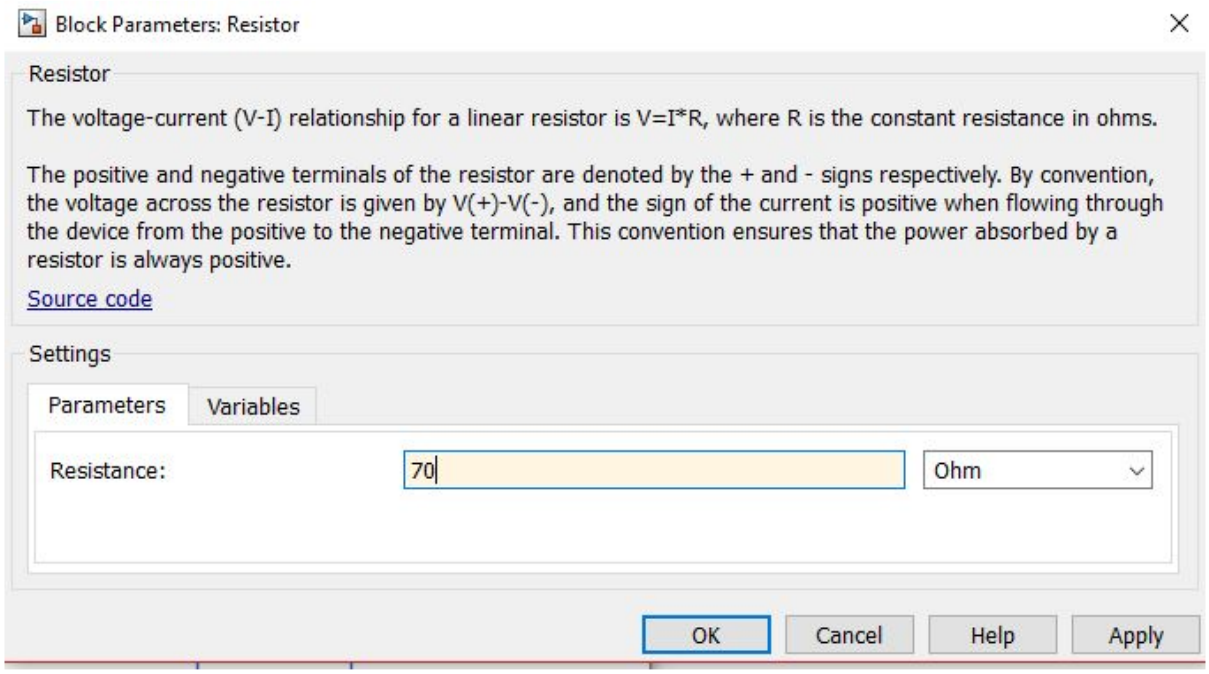

Obrázek 27: Skládání modelu RLC obvodu ze SimScape komponent

Nyní z knihovny *SimScape → Foundation Library → Electrical → Electrical Sources* přidáme blok *Controlled Voltage Source*. Poté z knihovny *SimScape → Foundation Library → Utilities* přidáme blok *Solver Configuration*. Zdroj napětí připojíme sériově ke třem již zapojeným blokům a obvod uzavřeme. K obvodu připojíme uzemnění a Solver Configuration (viz obrázek 28).



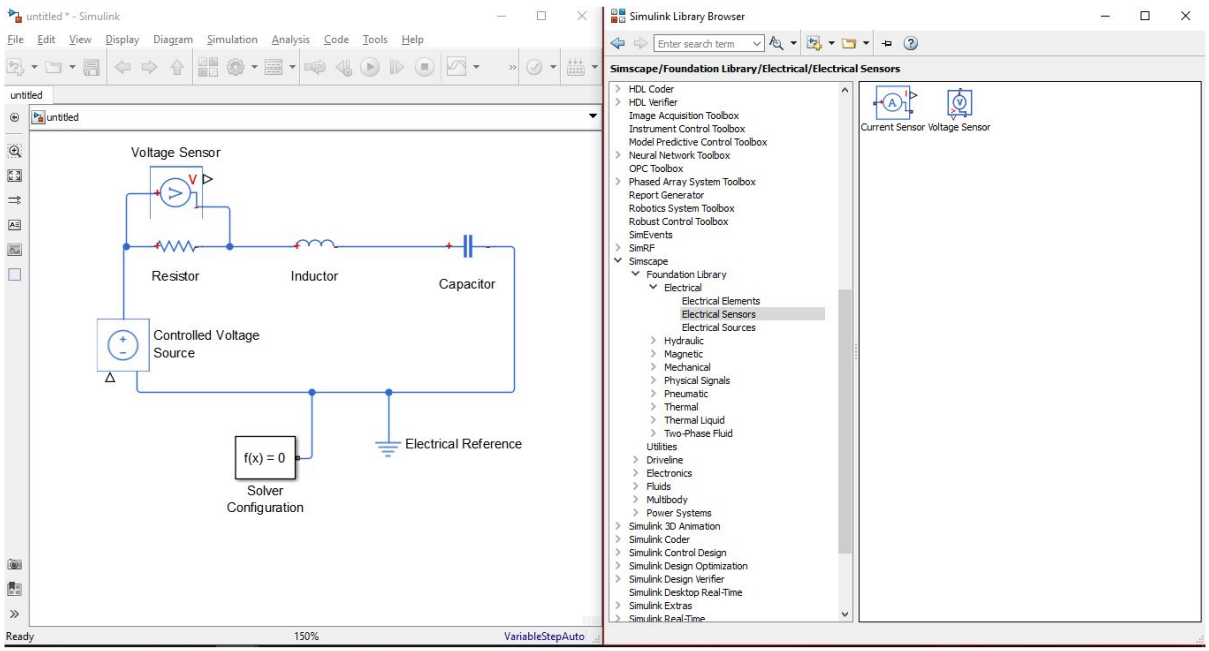

Obrázek 28: Přidávání nastavení solveru

Budeme potřebovat nastavit vhodné parametry pro fyzikální komponenty obvodu. Dvojklikem na blok otevřeme okno pro nastavení parametrů (viz obrázek 29).
Pro RLC obvod se osvědčily následující hodnoty:
odpor rezistoru $R$ = 70 Ω
indukčnost cívky $L$ = 0.04 H
kapacita kondenzátoru $C$ = 0.00003 F



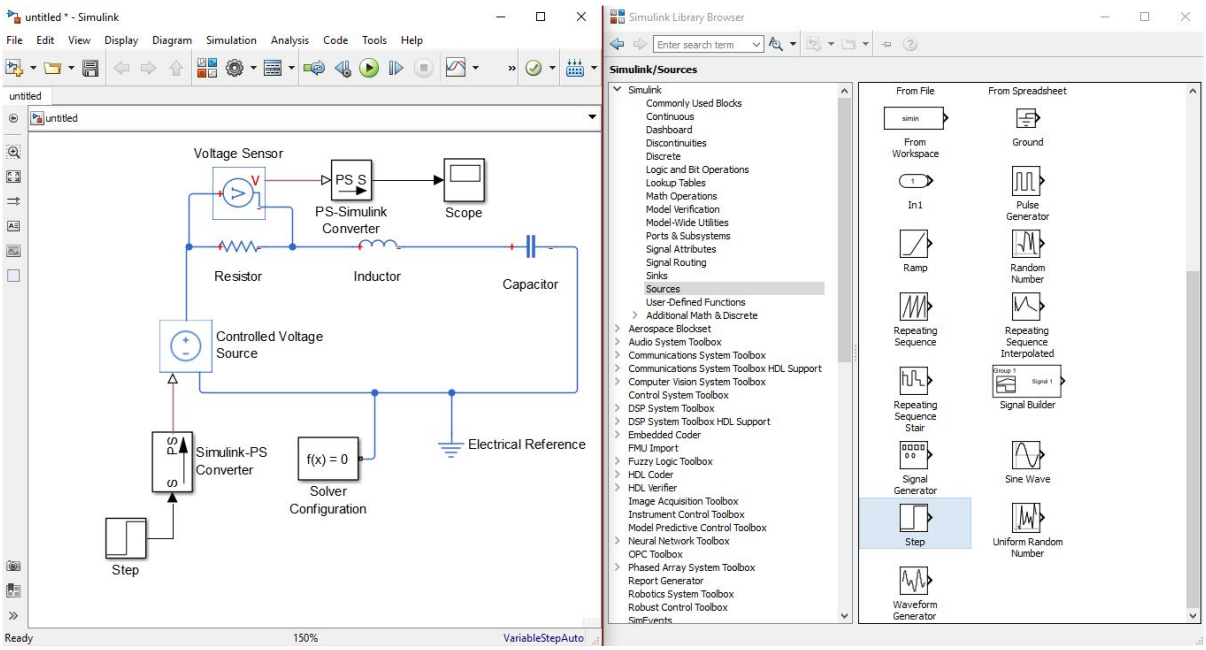

Obrázek 29: Nastavování parametrů jednotlivých bloků

Pro měření probíhajících napětí použijeme blok *Voltage Sensor* z knihovny *SimScape → Foundation Library → Electrical → Electrical Sensors,* připojíme ho paralelně k některému bloku, čímž budeme měřit napětí na tom daném bloku (viz obrázek 30). Pokud bychom chtěli měřit proud, použijeme z této knihovny blok *Current Sensor* a zapojíme ho sériově do obvodu (v nevětveném obvodu je proud na všech místech stejný).




Obrázek 30: Přidávání senzoru napětí

Senzory vydávají fyzikální signál (nikoliv pouze informační, který je běžný v Simulinku), který se na informační převede blokem z knihovny *SimScape → Foundation Library → Utilities* , *PS-Simulink Converter*. Výstupem tohoto bloku je signál použitelný pro základní komponenty v Simulinku, například *Scope* nebo *To Workspace*. Obdobně blok *Simulink-PS Converter* převede informační signál na fyzikální, a tak tento blok použijeme pro řízení zdroje napětí obvodu (viz obrázek 31). Nyní stačí dodat vstupní funkci pro zdroj napětí, nějakým způsobem uložit/zpracovat výstupy ze senzorů a nastavit parametry odporu, cívky a kondenzátoru (je možné odkázat se na proměnnou ve Workspace v MATLABu).



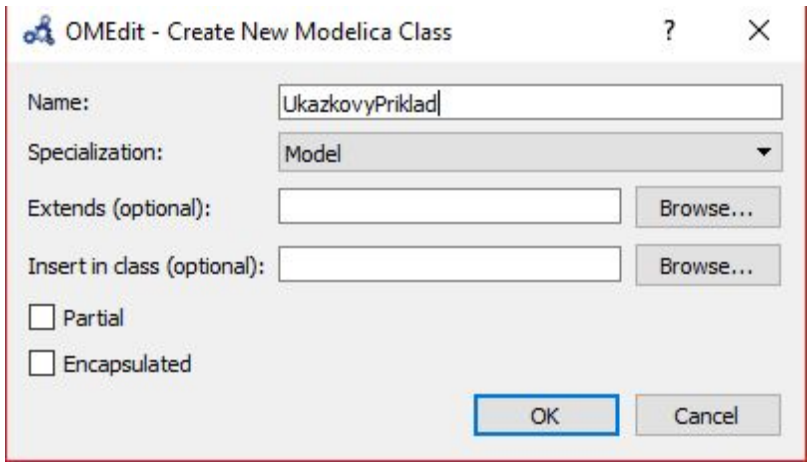

Obrázek 31: Poslední kroky při vytváření modelu RLC obvodu

# Postup pro OpenModelicu

V tomto postupu se naučíme vytvořit funkční blok, rovnou ho použijeme v interakci s jiným blokem a nakonec exportujeme model do formátu FMU.

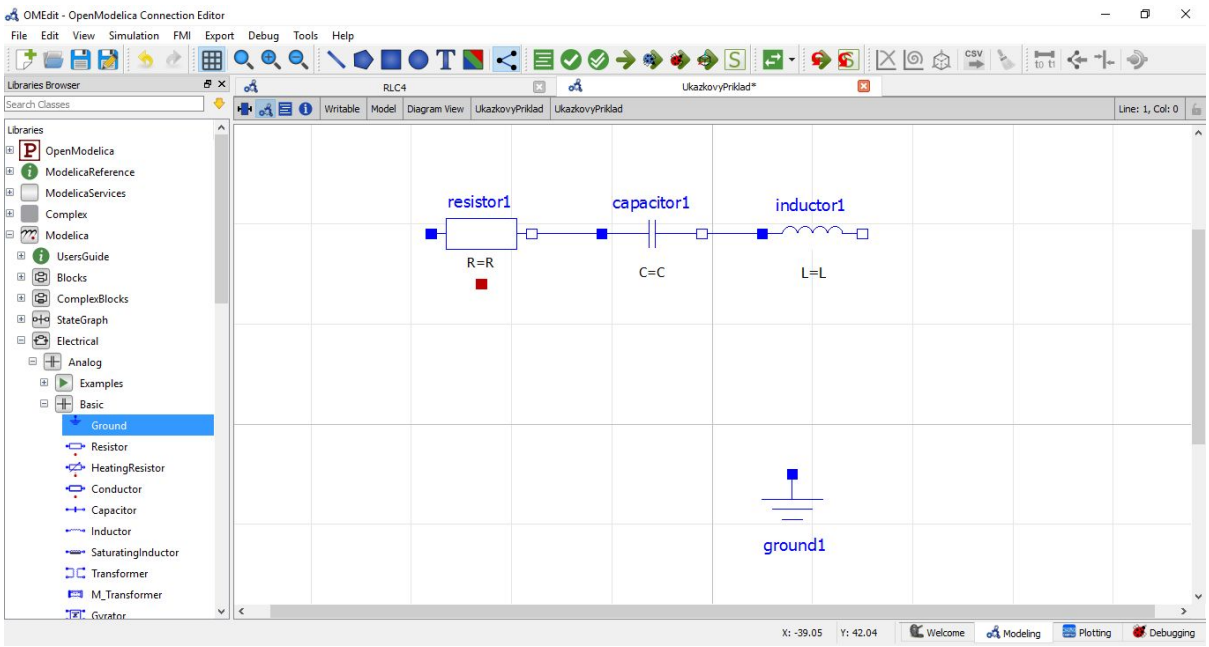
Začneme s hlavními prvky obvodu, budeme pokračovat přidáním měřících komponentů a nakonec dodáme interface pro ovládání zdroje.

Založíme nový prázdný model (viz obrázek 32) a do *diagam view* budeme přidávat bloky podobně jako v Simulinku.



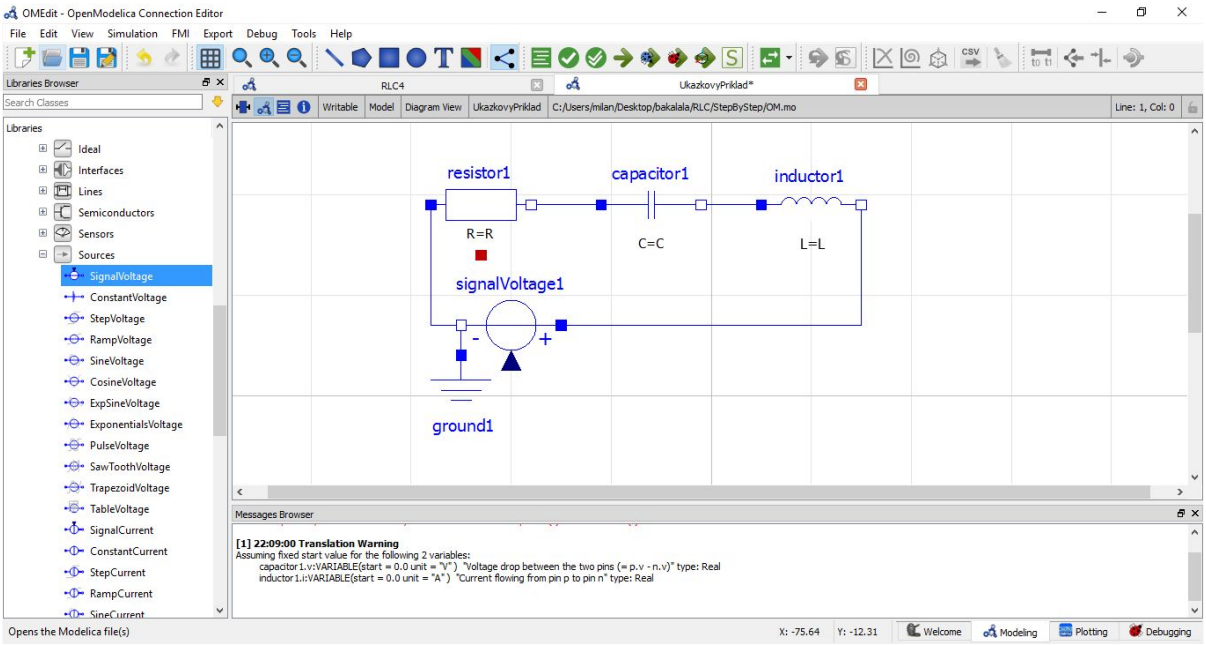

Obrázek 32: Vytváření modelu v OpenModelice

Z knihovny *Modelica → Electrical → Analog → Basic* přidáme do modelu bloky *Capacitor, Inductor, Resistor* a *Ground*. První tři zmíněné bloky zapojíme sériově jako na obrázku 33 níže.




Obrázek 33: Základní komponenty RLC obvodu

Nyní z knihovny *Modelica → Electrical → Analog → Sources* přidáme blok *SignalVoltage*. Zdroj napětí připojíme sériově ke třem již zapojeným blokům a obvod uzavřeme. K obvodu připojíme uzemnění. Na rozdíl od Simulinku zde nemůžeme vytvořit elektrický uzel, nicméně můžeme připojit více vodičů k jednomu pólu, a to také uděláme (viz obrázek 34).



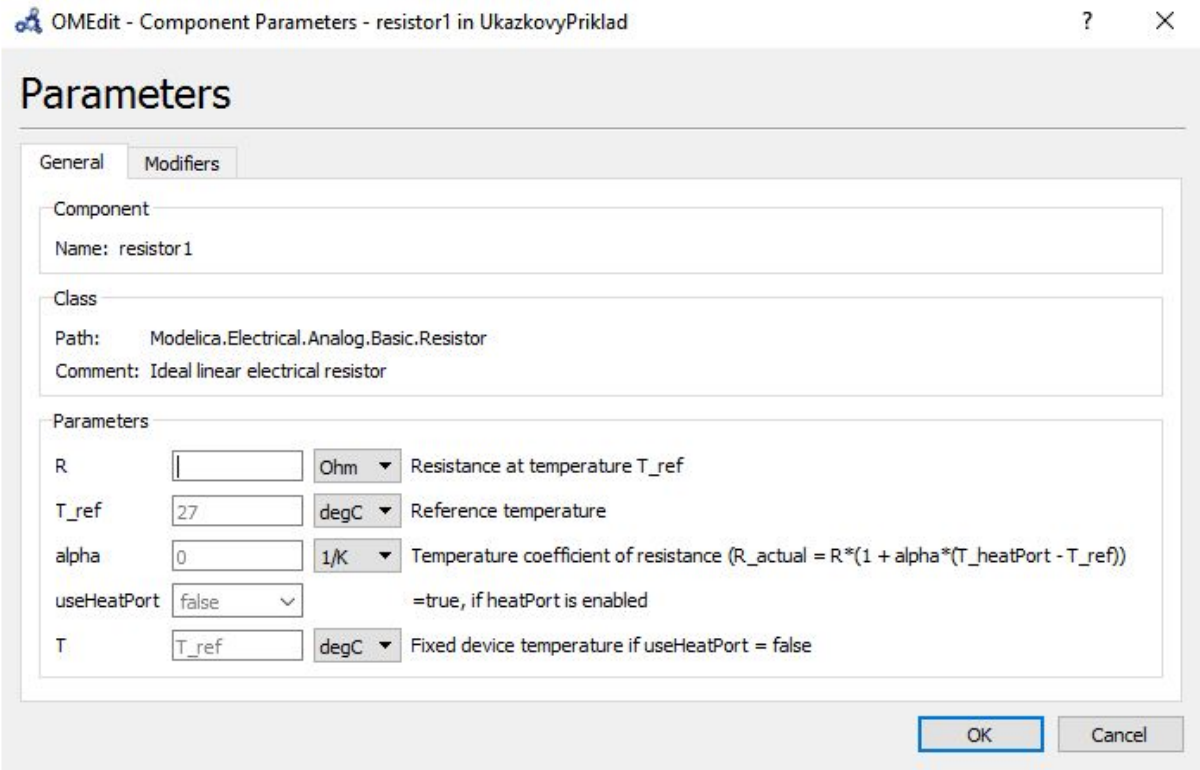

Obrázek 34: Přidání zdroje napětí v OpenModelice

Budeme potřebovat nastavit vhodné parametry pro prvky obvodu. Podobně jako v Simulinku se dvojklikem otevře okno pro nastavení parametrů prvku (viz obrázek 35). Pro RLC obvod se osvědčily následující hodnoty (důležité je mít stejné hodnoty jako u modelu v SimScape pro porovnání):

odpor rezistoru $R$ = 70 Ω

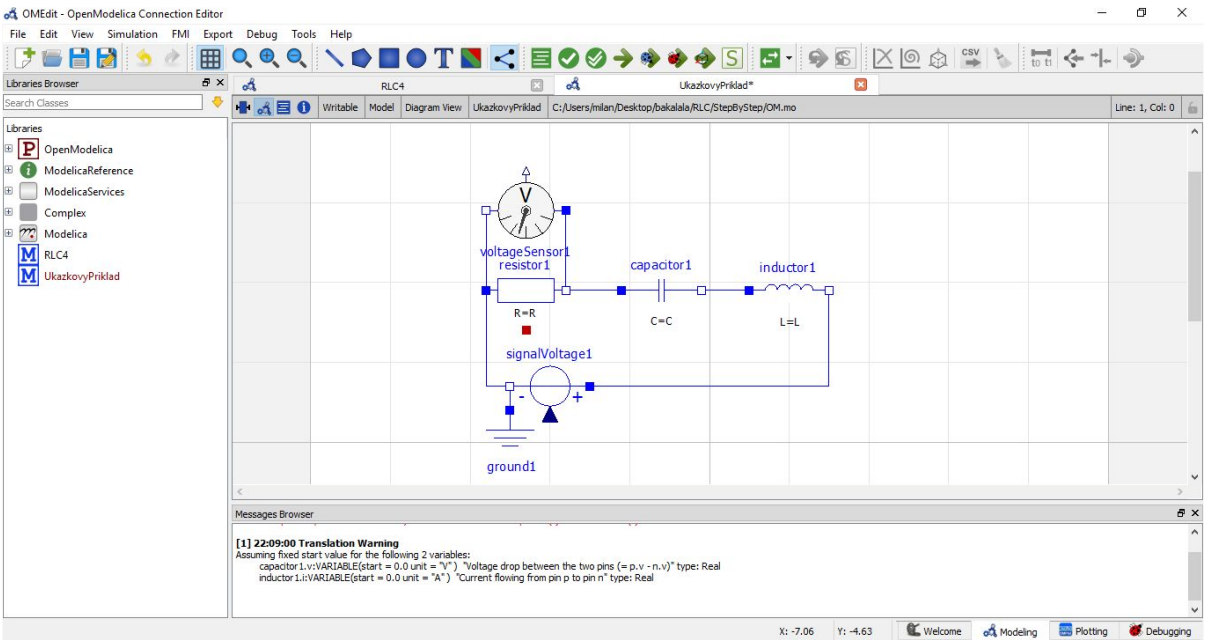
indukčnost cívky $L$ = 0.04 H

kapacita kondenzátoru $C$ = 0.00003 F



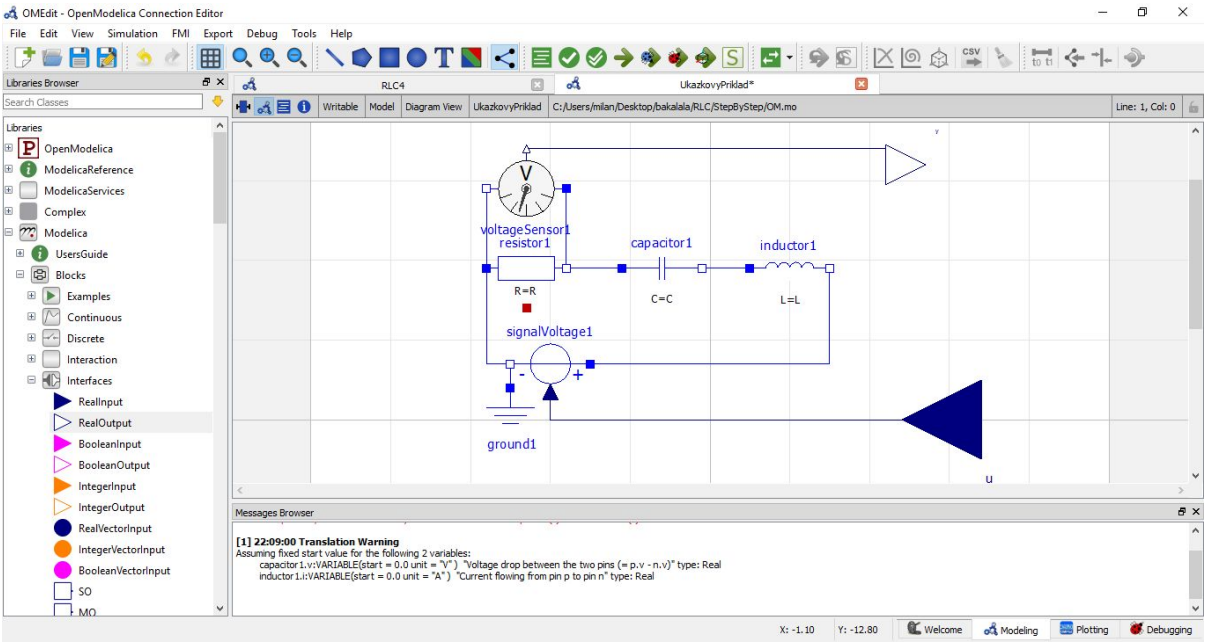

Obrázek 35: Nastavení parametrů bloku v OpenModelice

Pro měření probíhajících napětí použijeme z knihovny *Modelica → Electrical → Analog → Sensors* blok *VoltageSensor*, připojíme ho paralelně k některému bloku a tím budeme měřit napětí na tom daném bloku. Pokud bychom chtěli měřit proud, použijeme z této knihovny blok *CurrentSensor* a zapojíme ho sériově do obvodu (viz obrázek 36). Měření v OpenModelice tímto způsobem není nutné, neboť jsou průběhy všech veličin uloženy, ale výstup ze senzoru využijeme později.



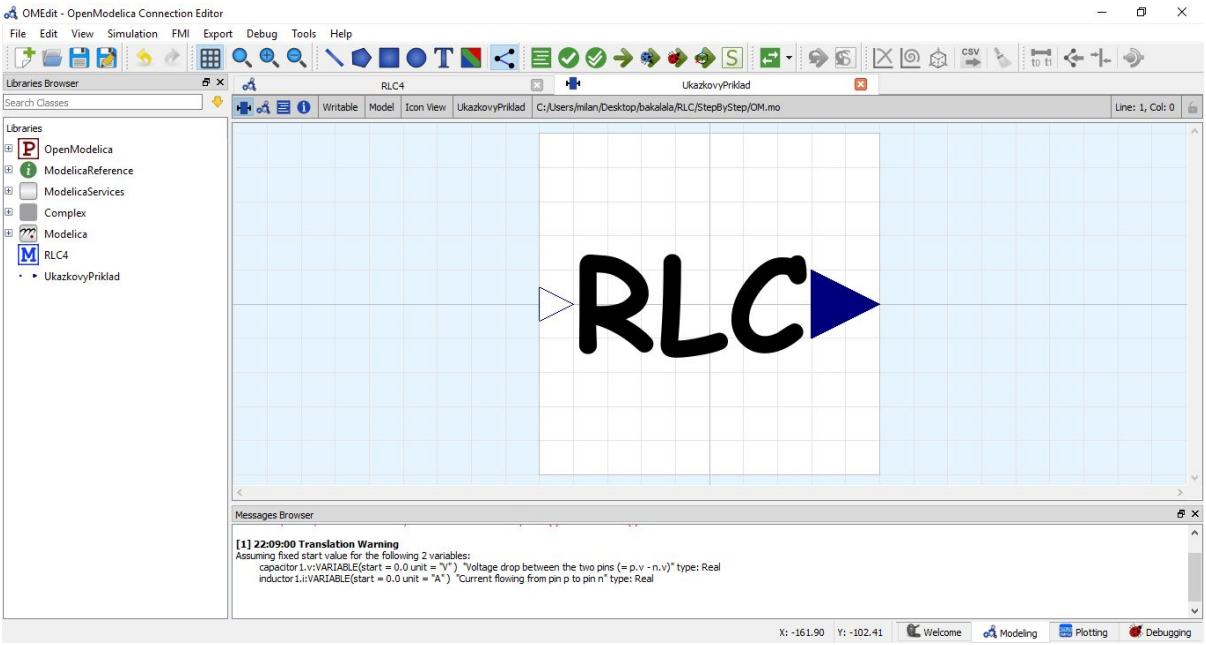

Obrázek 36: Přidání senzoru napětí

Nyní je třeba zajistit komunikaci tohoto bloku s ostatními. Na to použijeme bloky z knihovny *Modelica → Blocks → Interfaces*, a to bloky *RealInput* a *RealOutput*. *RealInput* připojíme ke vstupu zdroje napětí a *RealOutput* připojíme k senzoru napětí (viz obrázek 37).




Obrázek 37: Přidání interface pro ovládání napětí a snímání napětí

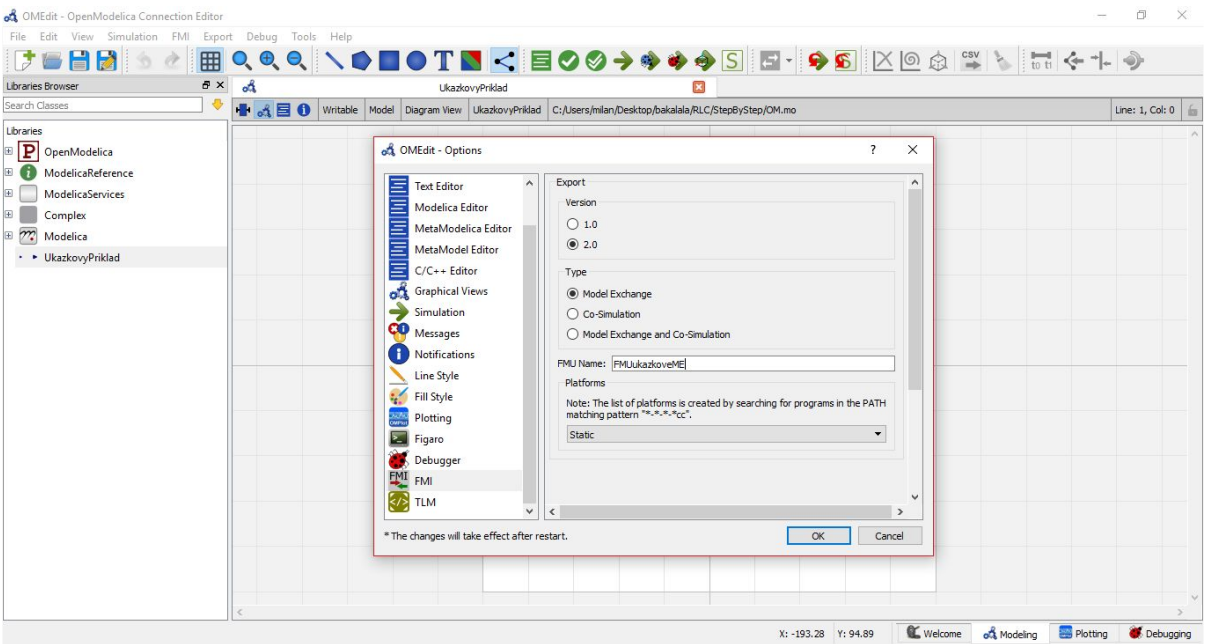
V *Icon view* vytvoříme požadovaný vzhled bloku a tím vytváření bloku končí (viz obrázek 38).




Obrázek 38: Vytváření vzhledu bloku

## Vytvoření FMU souboru

Nyní vytvoříme z modelu v OpenModelice .fmu soubor, který později použijeme vSimulinku. V horní liště je tlačítko *FMU*, nicméně to nám neumožní vlastnosti exportovaného modelu nijak ovlivnit, toto nastavení je v *Tools → Options → FMI* (viz obrázek 39).

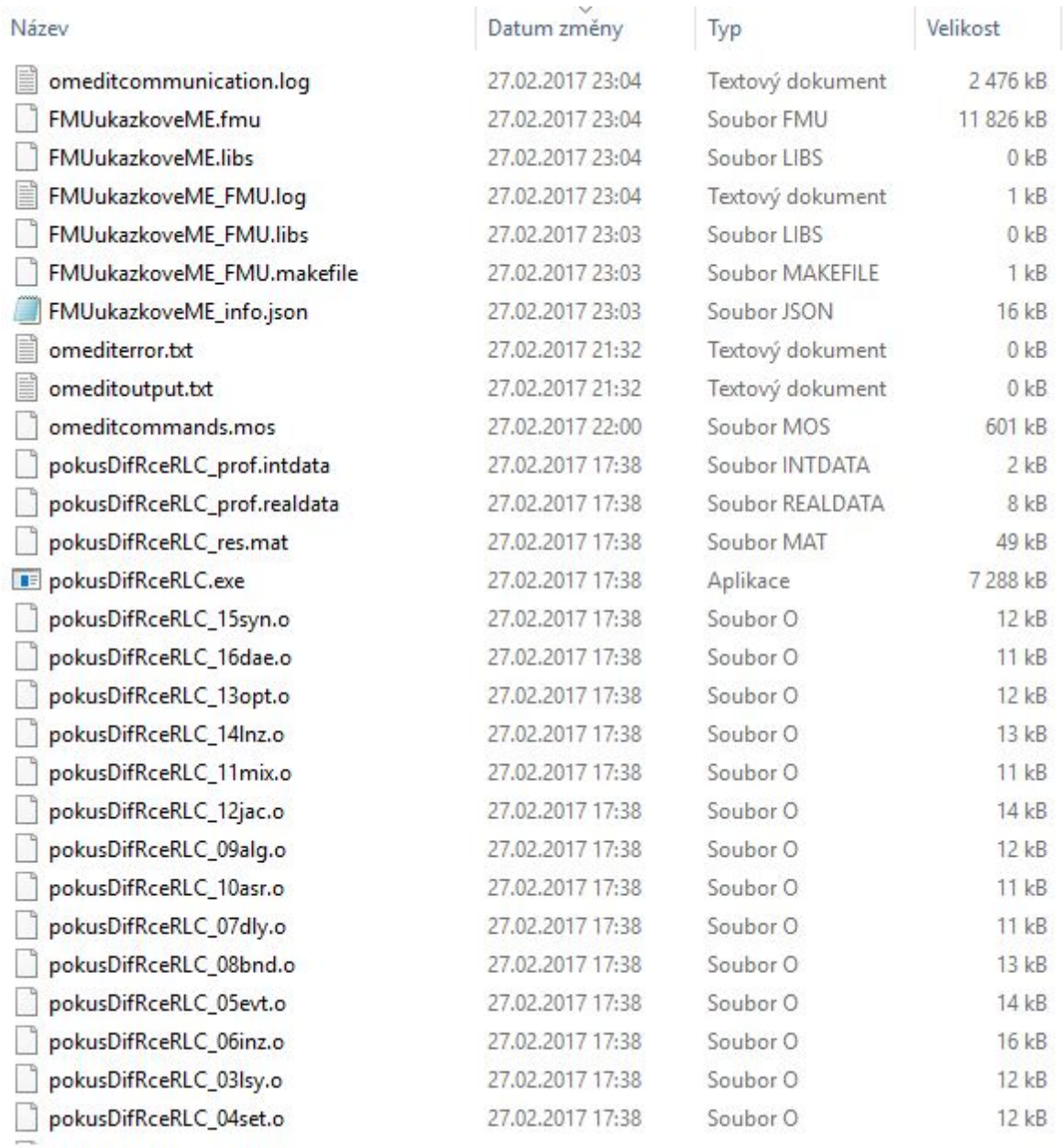



Obrázek 39: Nastavení exportu modelu jako FMU

Verzi FMU zvolíme aktuální (2.0), Type buď *Model Exchange* nebo *Co-Simulation*, pro porovnání budeme potřebovat oboje, je proto vhodné do názvu souboru zmínit, o kterou metodu jde. Možnost *Model Exchange and Co-Simulation* neumí Simulink momentálně zpracovat. Po nastavení všeho nutného exportujeme model do FMU přes možnost v horní liště. V message browseru (viz obrázek 40) se zobrazí cesta k exportovanému modelu.


Obrázek 40: Notifikační panel OpenModelicy po úspěšném exportování modelu

V cílové adrese je větší množství souborů (viz obrázek 41), nás ovšem zajímá pouze ten s příponou .fmu.

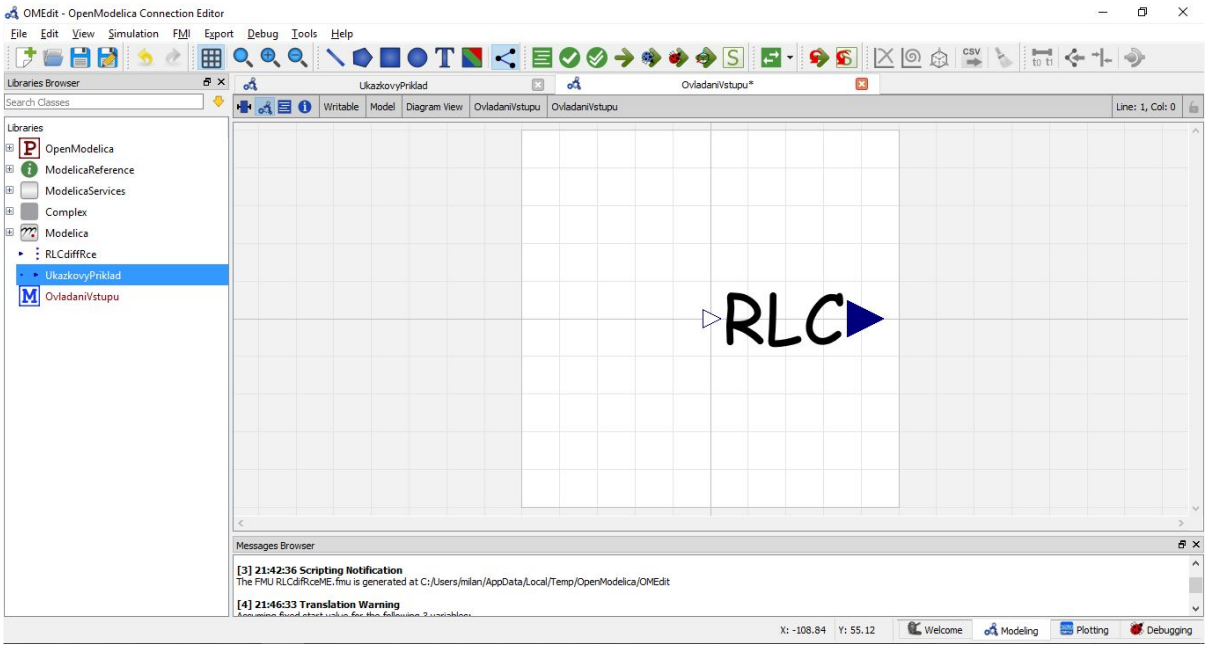


| Název | Datum změny | Typ | Velikost |
|---|---|---|---|
| omeditcommunication.log | 27.02.2017 23:04 | Textový dokument | 2 476 kB |
| FMUukazkoveME.fmu | 27.02.2017 23:04 | Soubor FMU | 11 826 kB |
| FMUukazkoveME.libs | 27.02.2017 23:04 | Soubor LIBS | 0 kB |
| FMUukazkoveME_FMU.log | 27.02.2017 23:04 | Textový dokument | 1 kB |
| FMUukazkoveME_FMU.libs | 27.02.2017 23:03 | Soubor LIBS | 0 kB |
| FMUukazkoveME_FMU.makefile | 27.02.2017 23:03 | Soubor MAKEFILE | 1 kB |
| FMUukazkoveME_info.json | 27.02.2017 23:03 | Soubor JSON | 16 kB |
| omediterror.txt | 27.02.2017 21:32 | Textový dokument | 0 kB |
| omeditoutput.txt | 27.02.2017 21:32 | Textový dokument | 0 kB |
| omeditcommands.mos | 27.02.2017 22:00 | Soubor MOS | 601 kB |
| pokusDifRceRLC_prof.intdata | 27.02.2017 17:38 | Soubor INTDATA | 2 kB |
| pokusDifRceRLC_prof.realdata | 27.02.2017 17:38 | Soubor REALDATA | 8 kB |
| pokusDifRceRLC_res.mat | 27.02.2017 17:38 | Soubor MAT | 49 kB |
| pokusDifRceRLC.exe | 27.02.2017 17:38 | Aplikace | 7 288 kB |
| pokusDifRceRLC_15syn.o | 27.02.2017 17:38 | Soubor O | 12 kB |
| pokusDifRceRLC_16dae.o | 27.02.2017 17:38 | Soubor O | 11 kB |
| pokusDifRceRLC_13opt.o | 27.02.2017 17:38 | Soubor O | 12 kB |
| pokusDifRceRLC_14lnz.o | 27.02.2017 17:38 | Soubor O | 13 kB |
| pokusDifRceRLC_11mix.o | 27.02.2017 17:38 | Soubor O | 11 kB |
| pokusDifRceRLC_12jac.o | 27.02.2017 17:38 | Soubor O | 14 kB |
| pokusDifRceRLC_09alg.o | 27.02.2017 17:38 | Soubor O | 12 kB |
| pokusDifRceRLC_10asr.o | 27.02.2017 17:38 | Soubor O | 11 kB |
| pokusDifRceRLC_07dly.o | 27.02.2017 17:38 | Soubor O | 11 kB |
| pokusDifRceRLC_08bnd.o | 27.02.2017 17:38 | Soubor O | 13 kB |
| pokusDifRceRLC_05evt.o | 27.02.2017 17:38 | Soubor O | 14 kB |
| pokusDifRceRLC_06inz.o | 27.02.2017 17:38 | Soubor O | 16 kB |
| pokusDifRceRLC_03lsy.o | 27.02.2017 17:38 | Soubor O | 12 kB |
| pokusDifRceRLC_04set.o | 27.02.2017 17:38 | Soubor O | 12 kB |

Obrázek 41: Složka vytvořená při generování FMU modelu

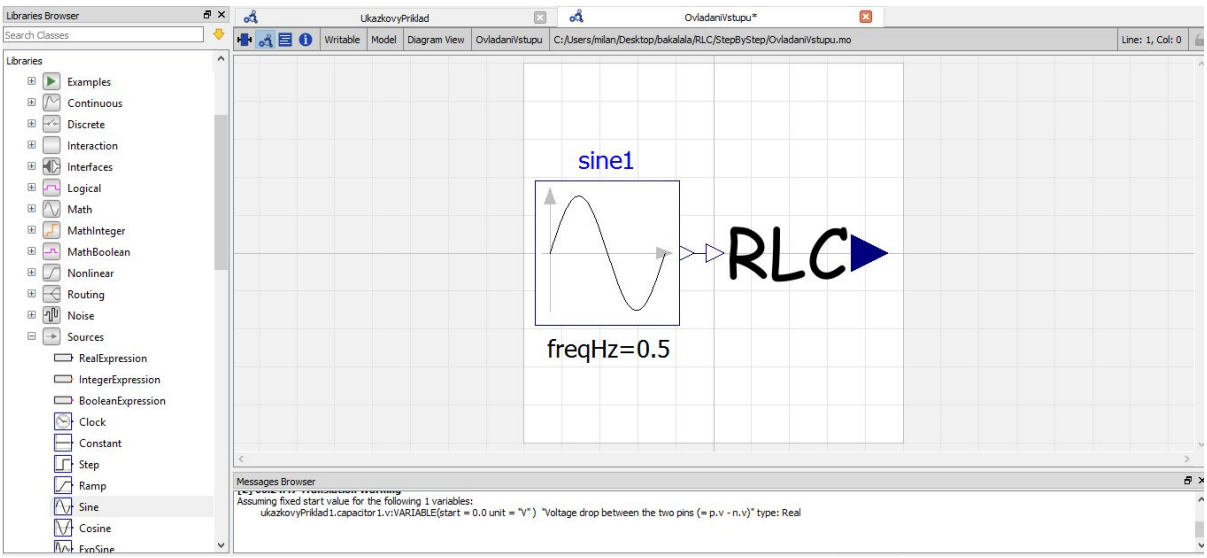
## Simulace bloku v OpenModelice

Nyní ještě zůstaneme v OpenModelice. Otevřeme nový model a vložíme do něj náš předchozí model (viz obrázek 42).



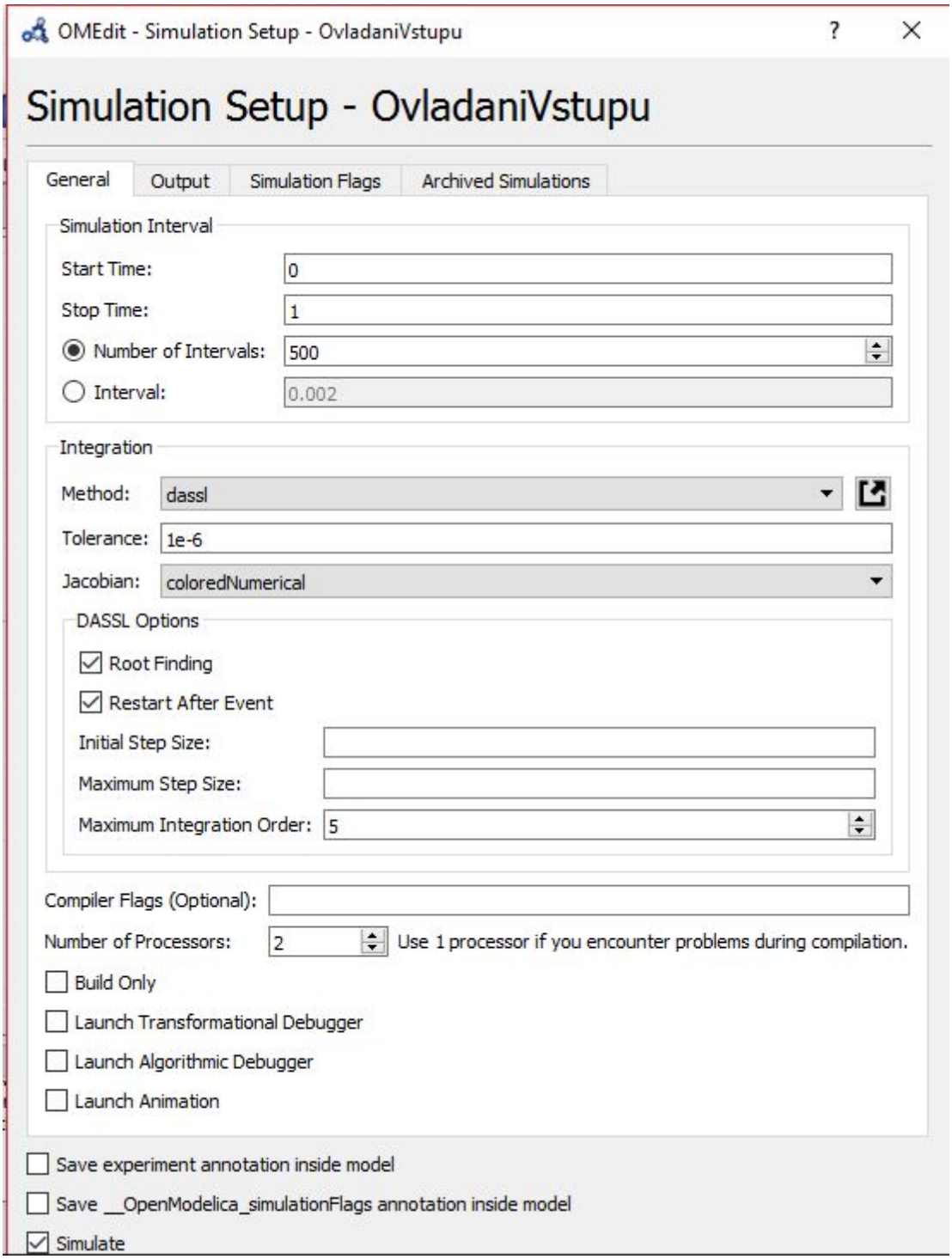

Obrázek 42: Použití vytvořeného modelu jako bloku

Nyní přidáme funkci, která bude určovat vstupní napětí do obvodu. Různé zdrojové funkce najdeme v knihovně *Modelica → Blocks → Sources* . Výstup zdroje připojíme na vstup našeho modelu (viz obrázek 43).



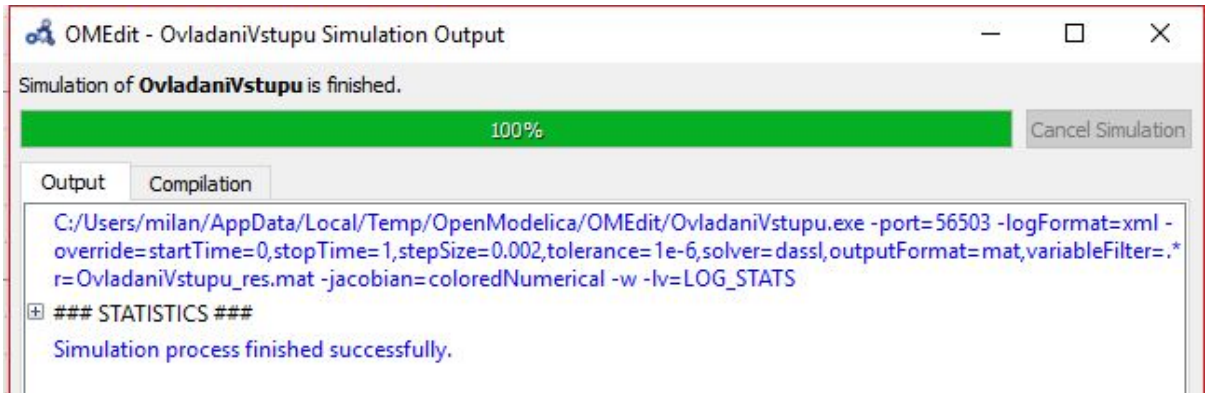

Obrázek 43: Interakce mezi bloky

Nyní tuto soustavu modelů odsimulujeme. Můžeme simulaci spustit tlačítkem *Simulate*. Pokud potřebujeme měnit vlastnosti simulace (doba simulace, vzorkovací perioda, použitý solver…), změníme nastavení v *Simulation Setup* (viz obrázek 44).



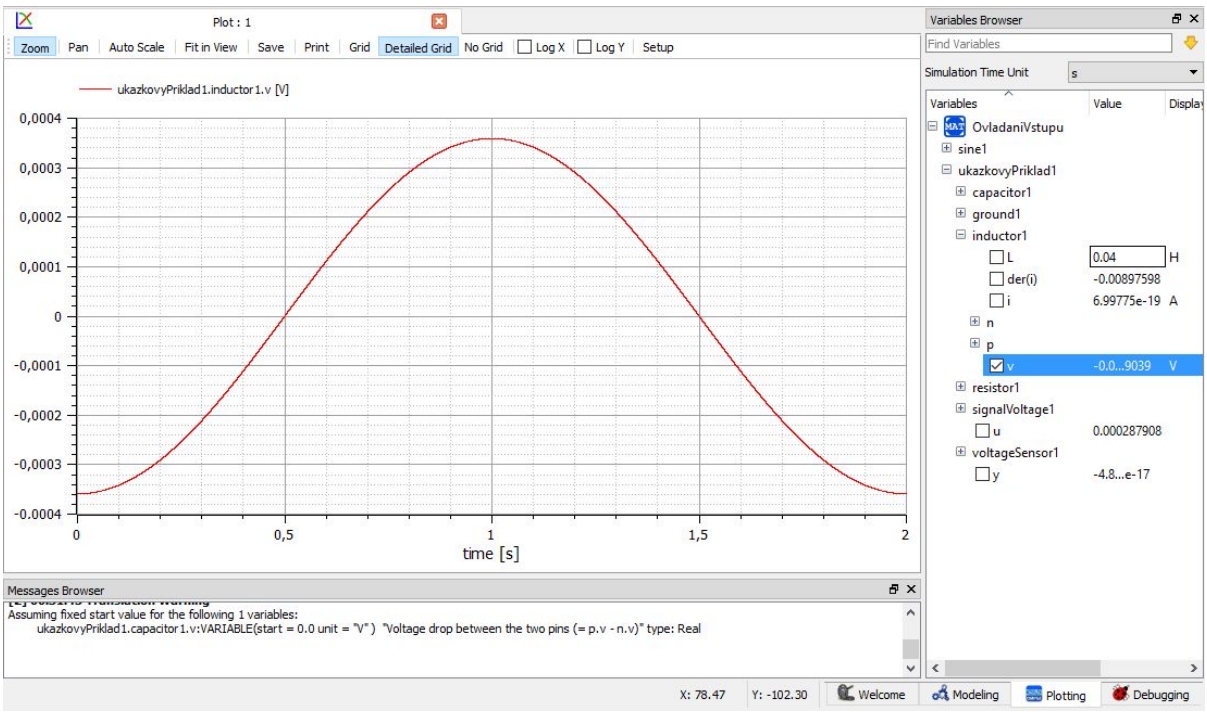

Obrázek 44: Nastavení simulace

Při simulování se objeví okno s průběhem kompilování modelu a stavu simulace (viz obrázek 45).




Obrázek 45: Okno zobrazené při simulování

Po úspěšném průběhu simulace můžeme v kartě *Plotting* vykreslit průběhy všech počítaných veličin (viz obrázek 46).

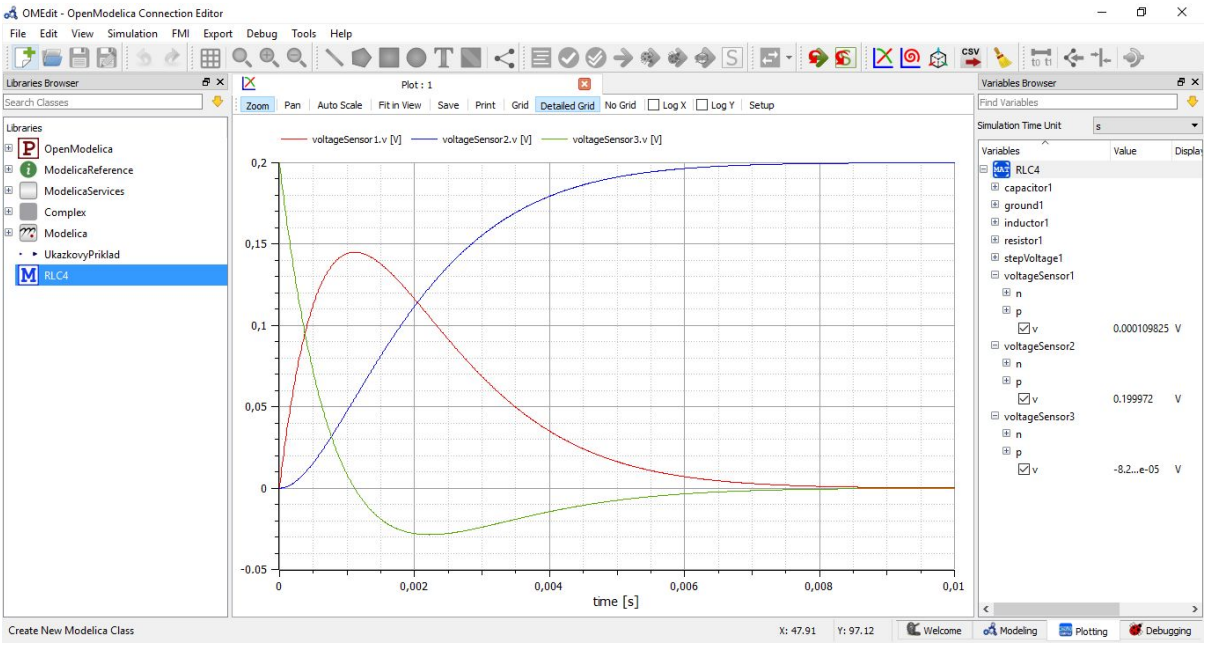



Obrázek 46: Zobrazení výstupu simulace

Samozřejmě je možné tuto soustavu bloků opět exportovat jako FMU.

## Export výsledků simulace z Modelicy do CSV

Nyní budeme exportovat výsledky simulace v Modelice do formátu .csv (Comma-Separated Values) a zobrazíme je v MATLABu. Výhoda tohoto postupu oproti Co-Simulation je, že k němu není potřeba knihovna v Simulinku podporující FMI.




Obrázek 47: Zobrazené veličiny se exportují

V Modelice stačí po odsimulování v kartě *Plotting* pouze zvolit vykreslené veličiny (např. jako na obrázku 47) a kliknout v horní liště na tlačítko 

, vybrat, kam se CSV soubor uloží a jak se bude jmenovat.

Soubor ve formátu CSV můžeme otevřít v široké škále programů. Jelikož jde o tabulkový formát, nabízí se Excel a jeho obdoby. Náhled na data je na obrázku 48:

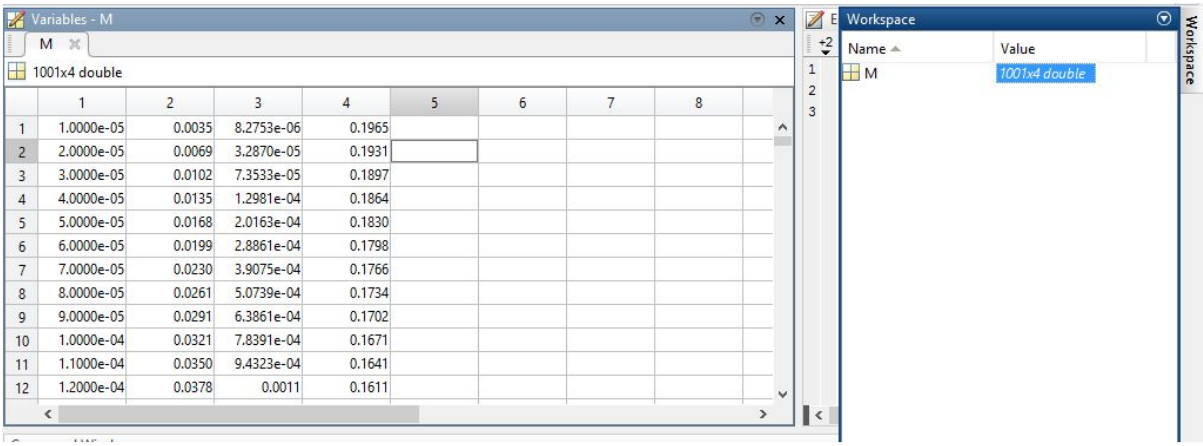
RLC_CSV.csv

| | A | B | C | D |
|---|---|---|---|---|
| 1 | time | voltageSensor1.v | voltageSensor2.v | voltageSensor3.v |
| 2 | 0 | 0 | 0 | 0.2 |
| 3 | 1.00E-05 | 0.00346953 | 8.28E-06 | 0.196522 |
| 4 | 2.00E-05 | 0.00687874 | 3.29E-05 | 0.193088 |
| 5 | 3.00E-05 | 0.0102283 | 7.35E-05 | 0.189698 |
| 6 | 4.00E-05 | 0.0135195 | 0.000129806 | 0.186351 |
| 7 | 5.00E-05 | 0.0167524 | 0.00020163 | 0.183046 |
| 8 | 6.00E-05 | 0.0199281 | 0.000288606 | 0.179783 |
| 9 | 7.00E-05 | 0.0230466 | 0.000390753 | 0.176563 |
| 10 | 8.00E-05 | 0.0261097 | 0.000507387 | 0.173383 |
| 11 | 9.00E-05 | 0.0291172 | 0.000638607 | 0.170244 |
| 12 | 0.0001 | 0.0320704 | 0.000783908 | 0.167146 |
| 13 | 0.00011 | 0.0349695 | 0.000943228 | 0.164087 |

Obrázek 48: Zobrazení exportovaných .csv dat v tabulkovém programu

Do MATLABu data z tohoto souboru načteme metodou csvread(). Jelikož nás zajímají pouze číselné hodnoty průběhů, načteme data od druhého řádku:

```
M=csvread('RLC_CSV.csv',2,0);
```

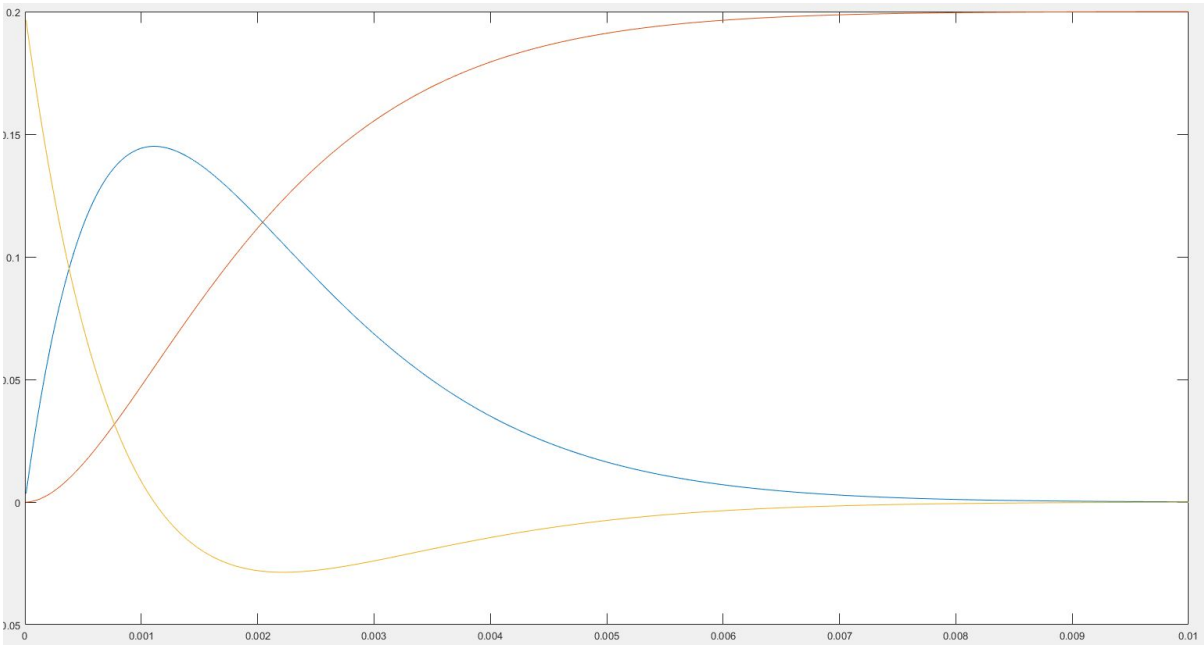



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1.0000e-05 | 0.0035 | 8.2753e-06 | 0.1965 | | | | |
| 2 | 2.0000e-05 | 0.0069 | 3.2870e-05 | 0.1931 | | | | |
| 3 | 3.0000e-05 | 0.0102 | 7.3533e-05 | 0.1897 | | | | |
| 4 | 4.0000e-05 | 0.0135 | 1.2981e-04 | 0.1864 | | | | |
| 5 | 5.0000e-05 | 0.0168 | 2.0163e-04 | 0.1830 | | | | |
| 6 | 6.0000e-05 | 0.0199 | 2.8861e-04 | 0.1798 | | | | |
| 7 | 7.0000e-05 | 0.0230 | 3.9075e-04 | 0.1766 | | | | |
| 8 | 8.0000e-05 | 0.0261 | 5.0739e-04 | 0.1734 | | | | |
| 9 | 9.0000e-05 | 0.0291 | 6.3861e-04 | 0.1702 | | | | |
| 10 | 1.0000e-04 | 0.0321 | 7.8391e-04 | 0.1671 | | | | |
| 11 | 1.1000e-04 | 0.0350 | 9.4323e-04 | 0.1641 | | | | |
| 12 | 1.2000e-04 | 0.0378 | 0.0011 | 0.1611 | | | | |

Obrázek 49: Zobrazení dat po naimportování v MATLABu

Nyní můžeme načtené veličiny vykreslit jako na obrázku 50.

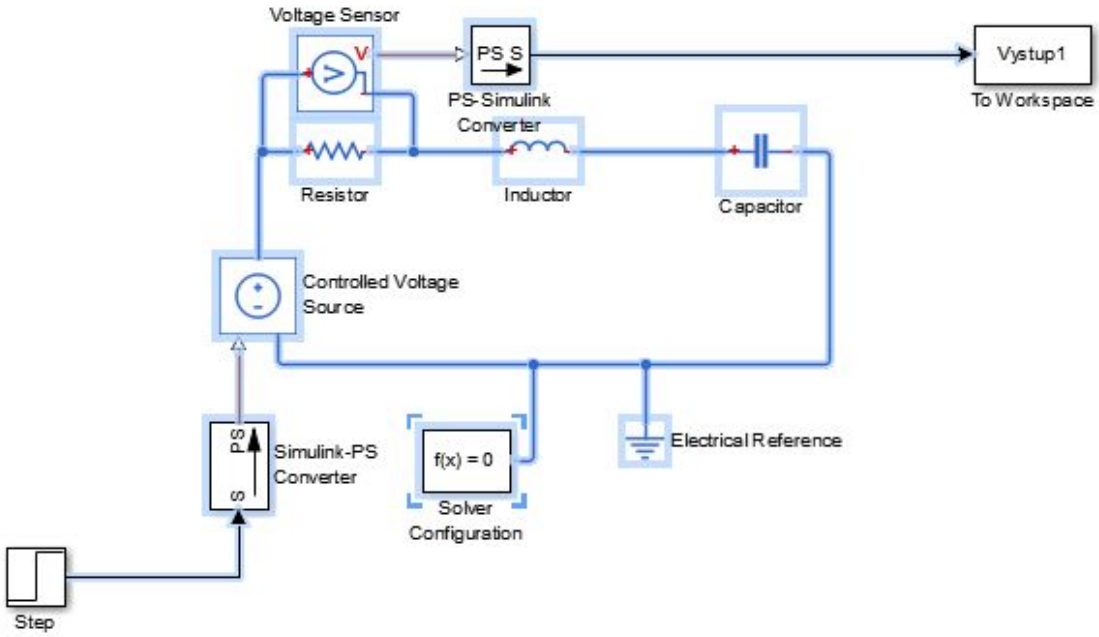
```
plot(M(:,1),M(:,2),M(:,1),M(:,3),M(:,1),M(:,4))
```



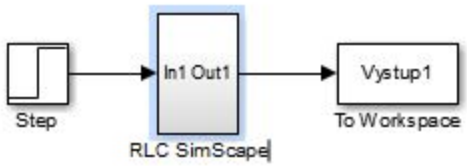

Obrázek 50: Vykreslení dat v MATLABu

## Porovnání FMU a SimScape modelu

Otevřeme si v Simulinku nový model a náš model ze SimScapu do něj zkopírujeme. Pro větší přehlednost uděláme z obvodu subsystém, třeba jako na obrázcích 51 a 52:

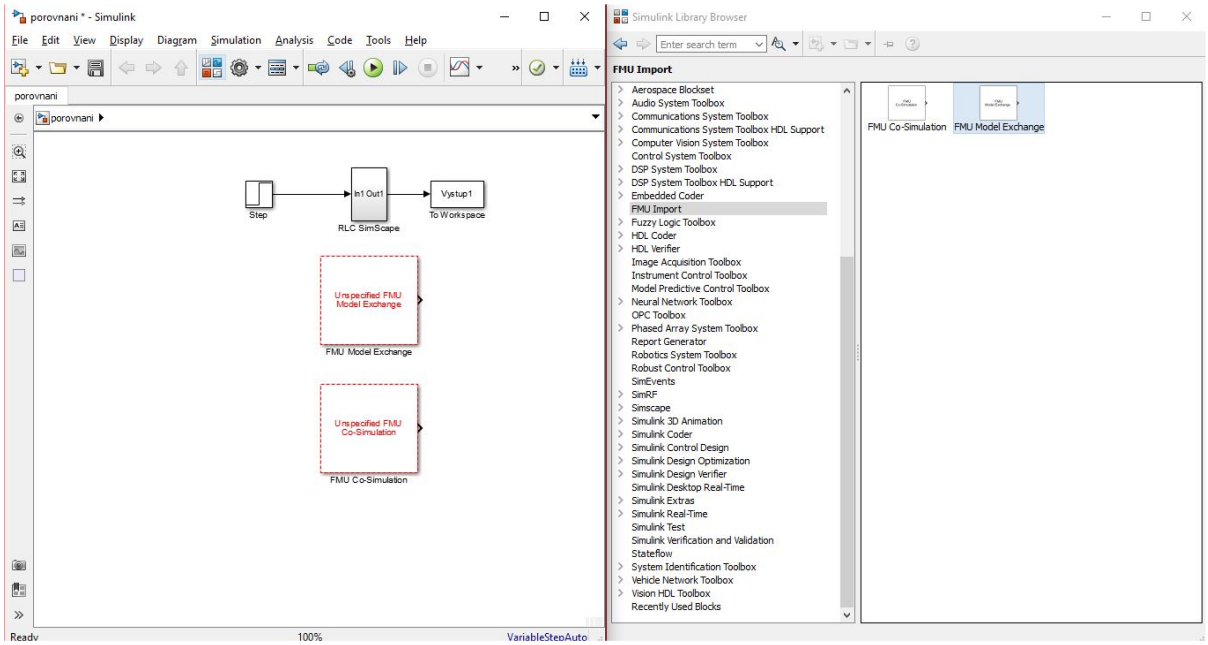



Obrázek 51: Vytváření subsystému z části modelu



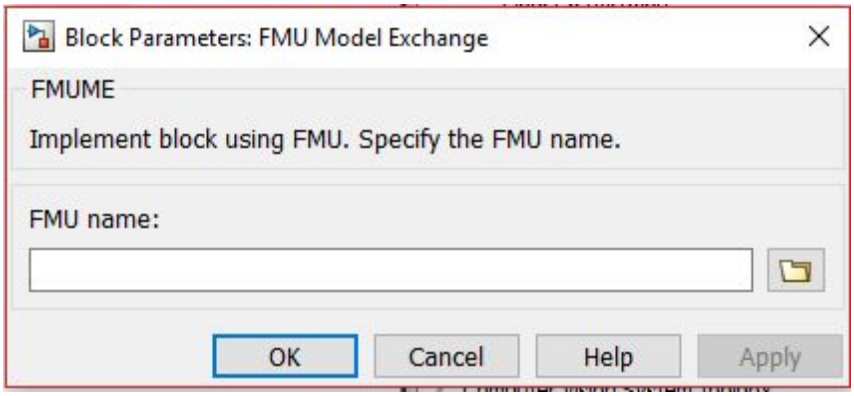

Obrázek 52: Vytvořený subsystém

Poté z knihovny *FMU Import* do modelu přidáme oba bloky (viz obrázek 53).



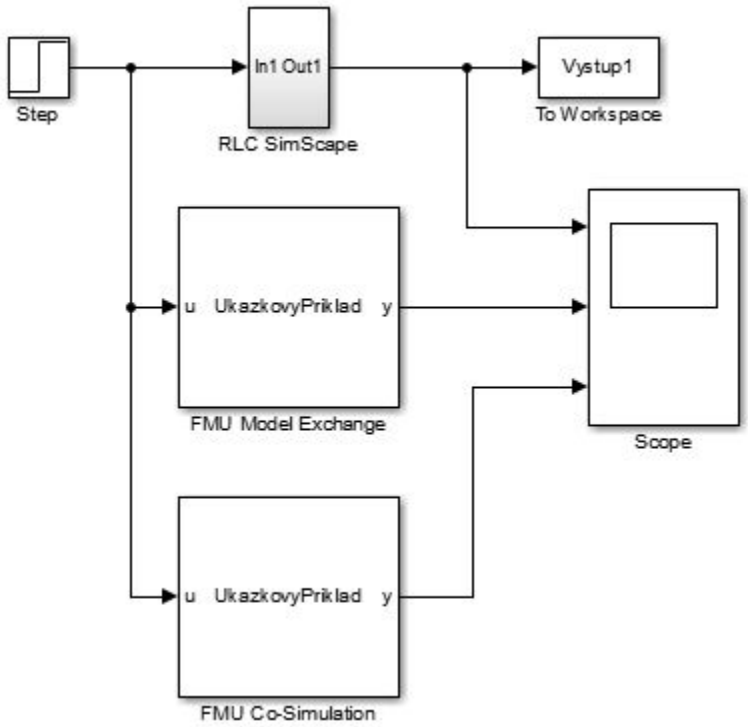

Obrázek 53: Bloky Pilot Support Package pro FMU Import

Při rozkliknutí bloku zvolíme cestu ke zvolenému .fmu souboru (viz obrázek 54):



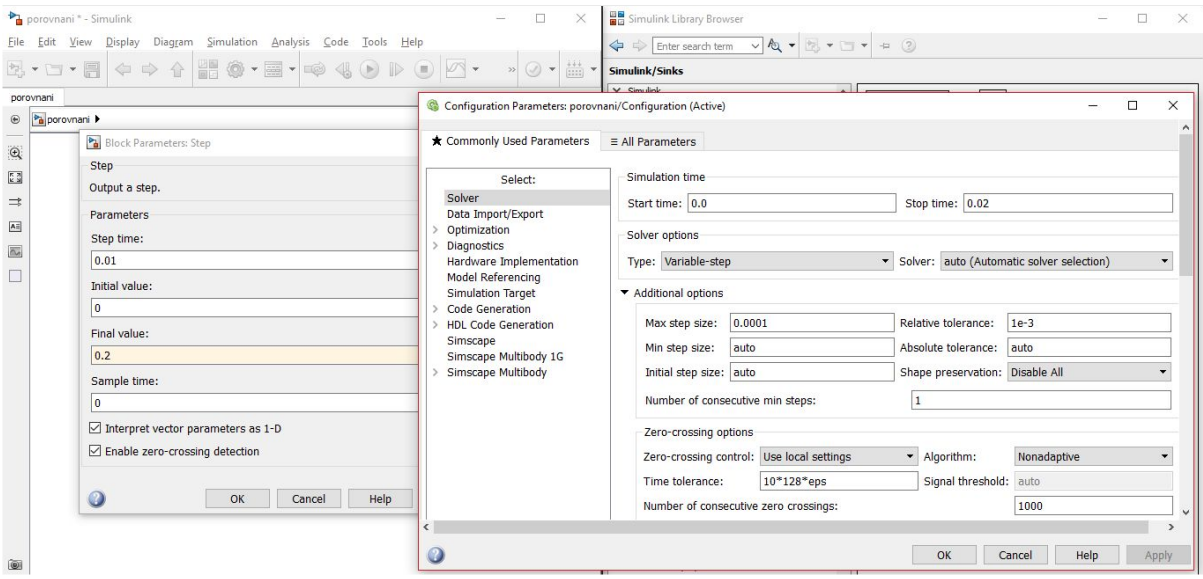

Obrázek 54: Nastavení bloku pro import FMU

Po nahrání FMU souboru spojíme zdrojovou funkci se vstupem a výstup zpracujeme, například připojíme na *Scope* nebo výstup uložíme do *Workspace* (např. jako na obrázku 55).



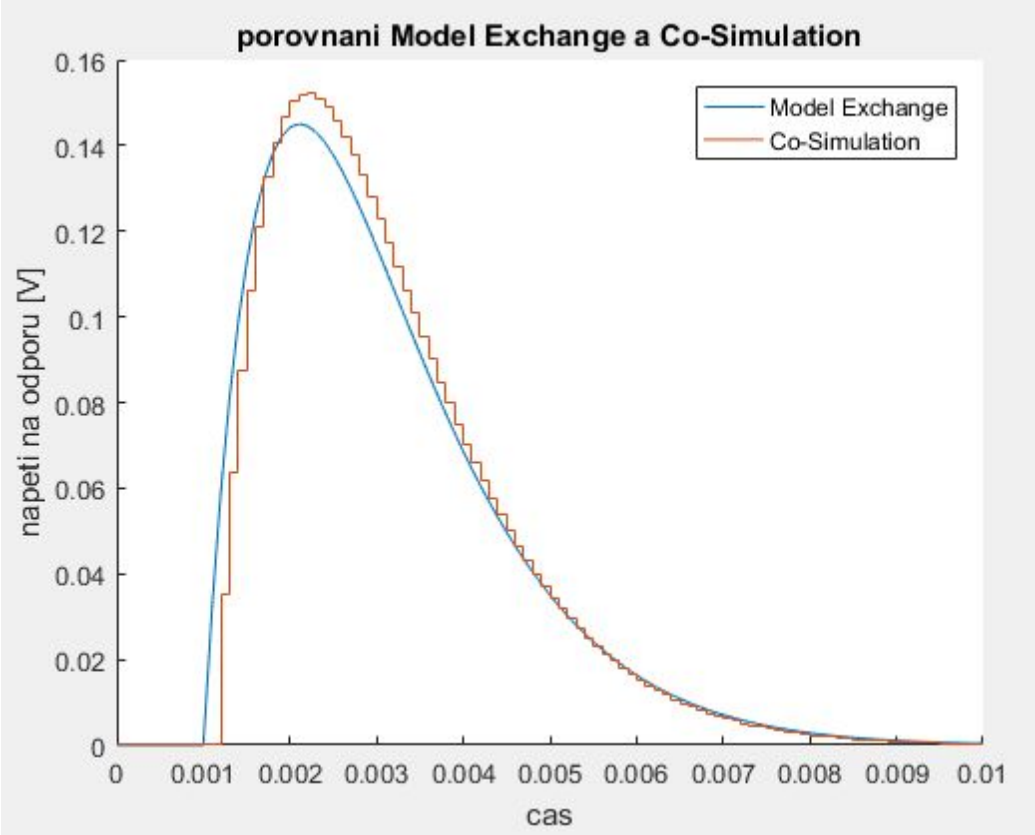

Obrázek 55: Schéma porovnání modelů

Pokud používáme jako vstup *Step*, bývá reakce na něj velmi rychlá. Abychom mohli pohodlně zkoumat průběh veličin, je vhodné zvolit dostatečně krátký časový interval a také snížit maximální délku kroku pro hladkost a přesnost průběhu (viz obrázek 56).



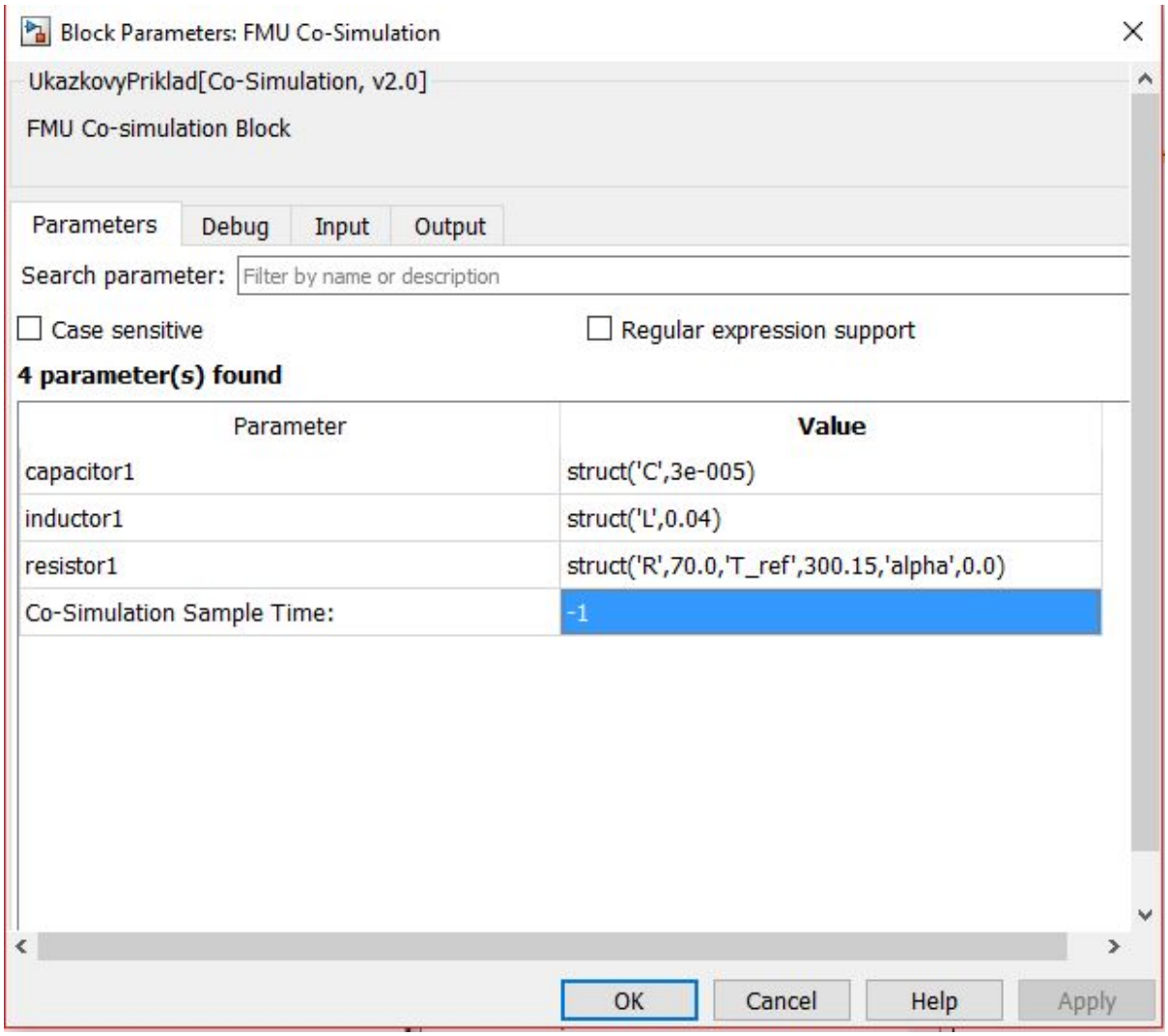

Obrázek 56: Nastavení parametrů simulace v Simulinku

Zatímco průběh napětí na odporu je u modelu v SimScapu a FMU Model Exchange naprosto totožný, chování Co-Simulation modelu je poněkud odlišné, mimo jiné i délkou kroku (viz obrázek 57).



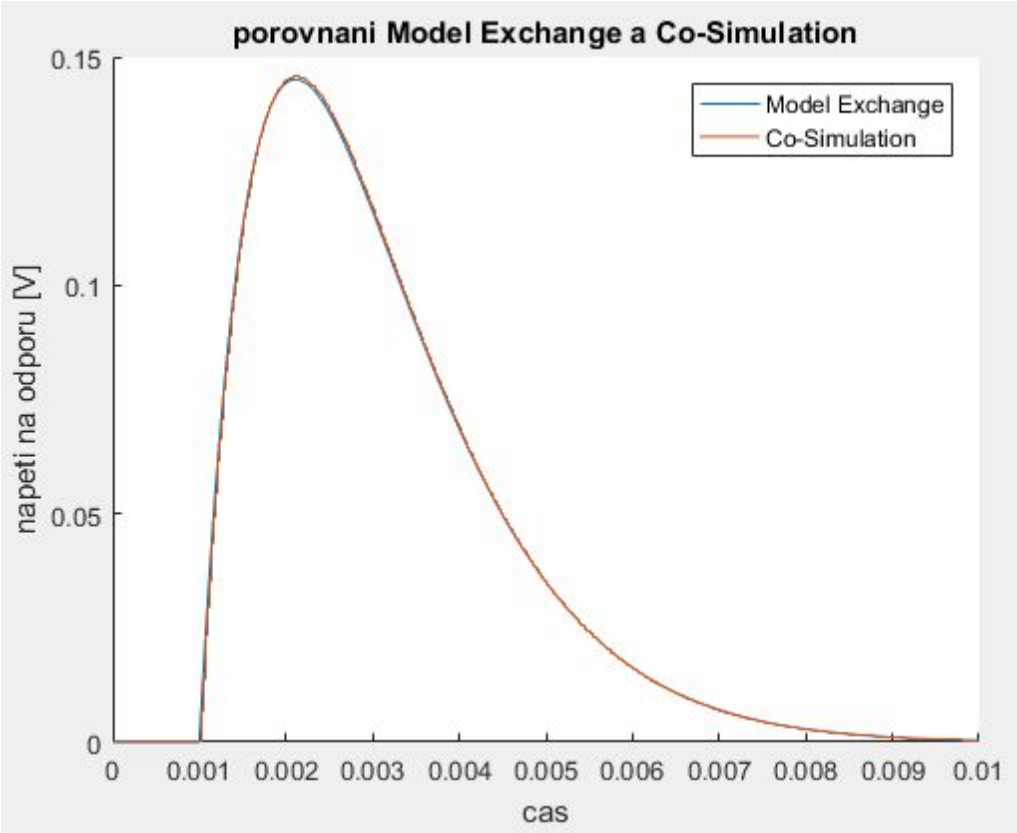

Obrázek 57: Znázornění rozdílu defaultního nastavení FMU Co-Simulation a FMU Model Exchange

Je zřejmé, že "děděný" Sample Time z původního modelu není totožný, proto ho zkusíme změnit na délku kroku, kterou jsme nastavili u solveru v Simulinku (viz obrázek 58).



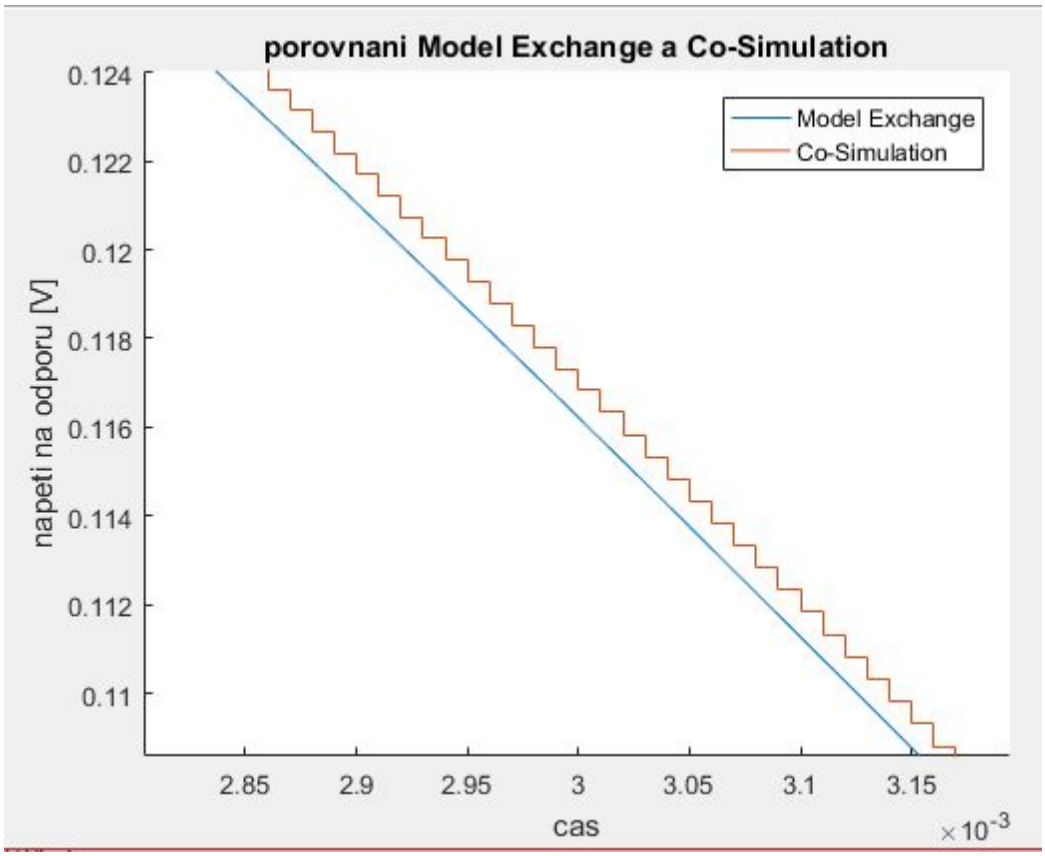

Obrázek 58: Nastavení FMU modelu

Tato změna se silně projevila na přesnosti výpočtu:




Obrázek 59: Průběh simulace s menší simulační periodou

Ovšem při detailnějším pohledu (viz obrázek 60) zjistíme, že průběhy se stále zcela neshodují, nehledě na Zero-order-hold u průběhu Co-Simulation.




Obrázek 60: Detail průběhu znázorněné veličiny

# Model Turbíny

## Základní rovnice

Tato část práce se bude zabývat převedením matematického modelu vodní turbíny a generátoru v prostředí Simulink do standardu FMU [13]. Poté dojde k vytvoření regulátoru s použitím modelu regulátoru ze systému REX, a následně budou tyto modely zprovozněny v systému reálného času REX a bude prověřena jejich přesnost oproti ekvivalentním modelům v prostřední MATLAB/Simulink.

Byl získán následující model vodní turbíny [14]:

$$U = G \cdot \sqrt{H}$$
$$P_m = A_t \cdot H \cdot (U - gnl) - K_d \cdot \Delta\omega \cdot G$$
$$\frac{dU}{dt} = \frac{H - H_0}{T_\omega}$$

kde $U$ je aktuální průtok, $G$ je pozice vstupního ventilu, $H$ je výška vodního sloupce, $P_m$ je mechanický výkon, $A_t$ je konstanta pro konkrétní systém, $gnl$ je mechanická ztráta proudění, $K_d$ je proporční koeficient, $\Delta\omega$ je změna otáček a $T_\omega$ je časová konstanta.

Model je vytvořen bez konkrétních jednotek, modeluje pouze vztahy mezi jevy, nikoliv konkrétní veličiny, protože při jeho tvorbě nebyla k dispozici data z reálného systému.

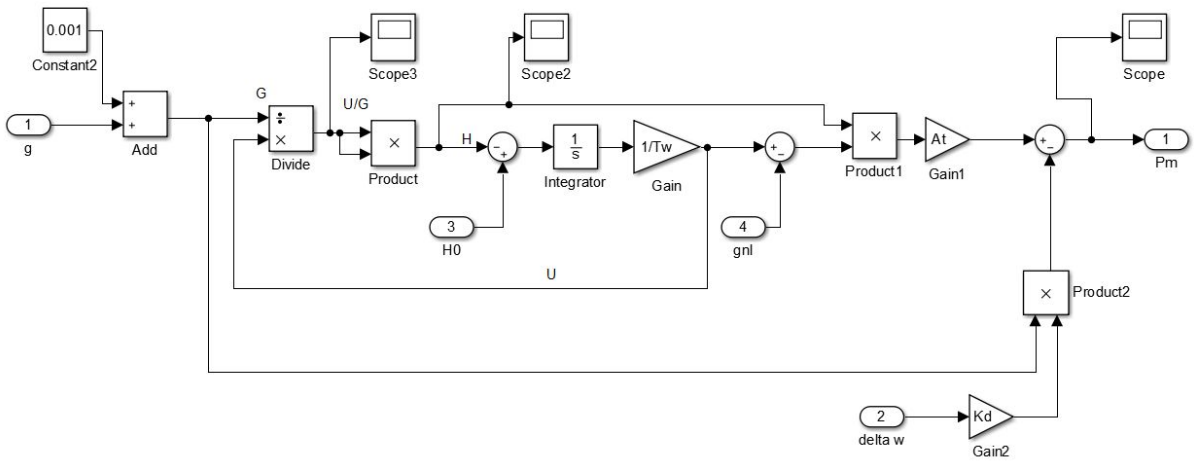
Model generátoru:

$$M \cdot \frac{d\Delta\omega}{dt} = P_m - P_e - P_d$$
$$\frac{d\delta}{dt} = \Delta\omega$$

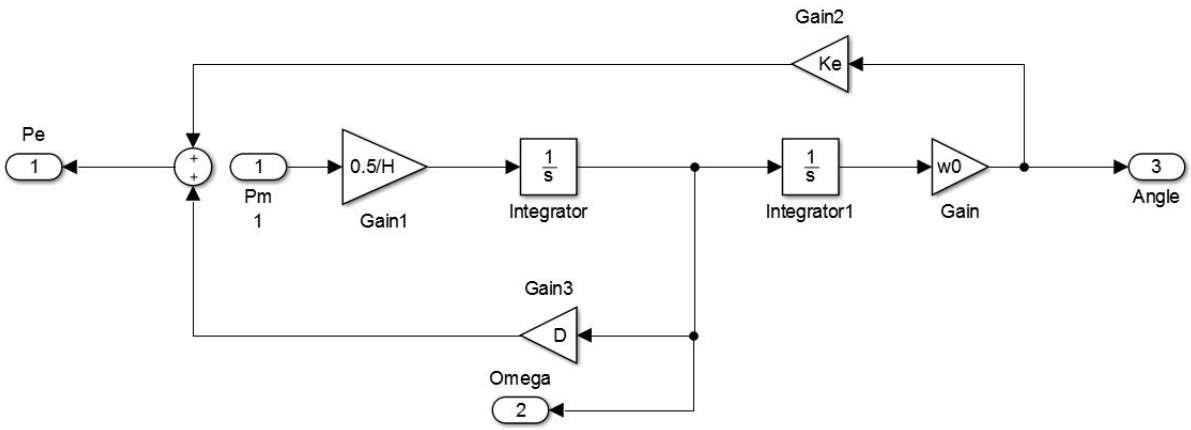
kde M je kroutící moment, $P_e$ je elektrický výkon, $P_d$ je tlumení a $\delta$ je úhel natočení rotoru.

# Modely v Simulinku, OpenModelice a REXu

Těmto modelům odpovídají modely v Simulinku na obrázcích 61 a 62.

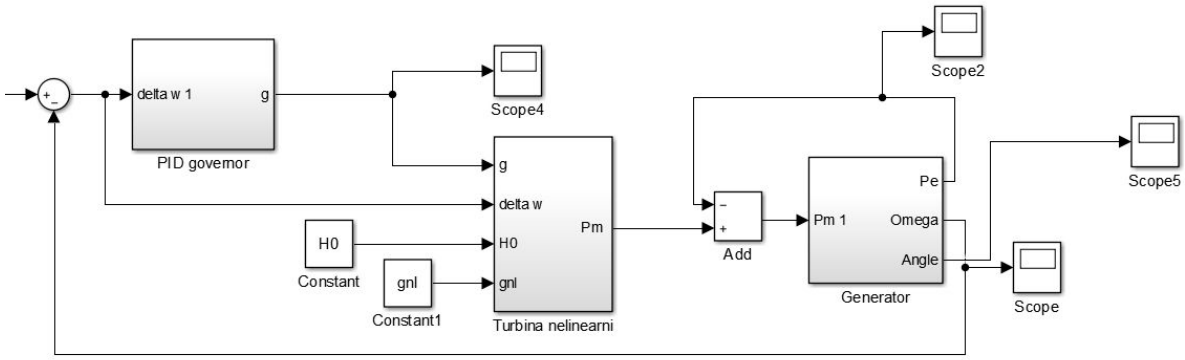



Obrázek 61: Model turbíny




Obrázek 62: Model generátoru

Celkové zapojení s jednoduchým PI regulátorem potom vypadá jako na obrázku 63.




Obrázek 63: Zapojení generátoru a turbíny s řízenými otáčkami

V Modelice byly na základě rovnic vytvořeny modely turbíny a generátoru, viz obrázky 64 a 65.

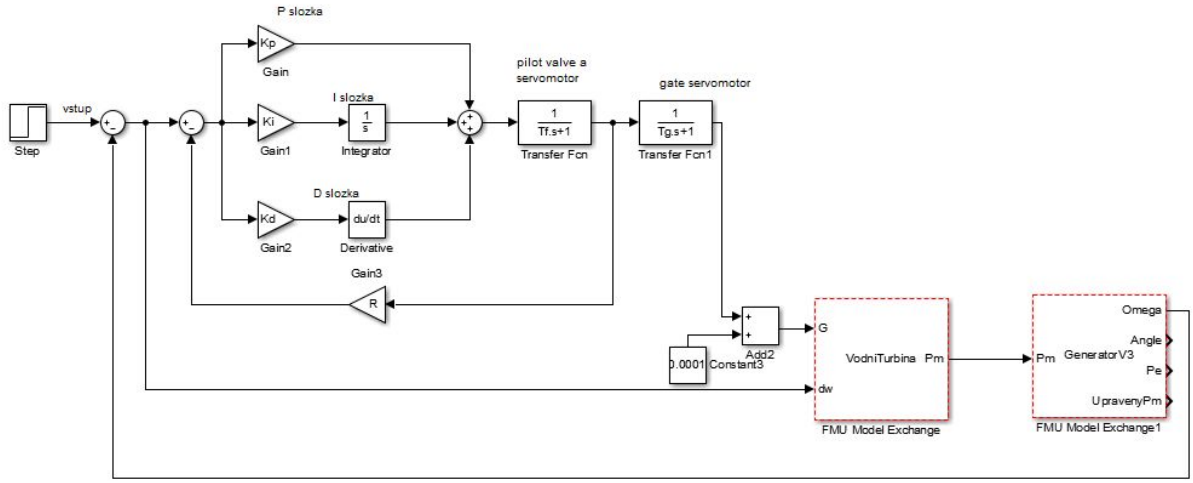
```
equation
  U=G*sqrt(H);
  Pm=At*H*(U-gnl)- (Kd*dw*G);
  der(U)=(H0-H)/Tw;
```

Obrázek 64: Zápis modelu turbíny v jazyce Modelica

```
UpravenyPm = Pm - Pe;
der(Omega) = UpravenyPm * 0.5 / H;
der(Angle1) = Omega;
Angle = Angle1 * w0;
Pe = Angle * Ke + Omega * D;
```

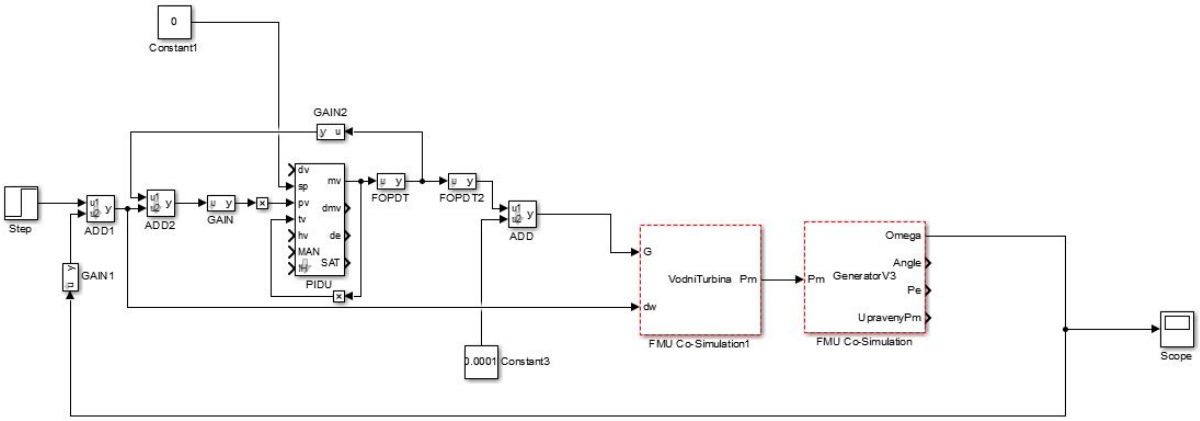
Obrázek 65: Zápis modelu generátoru v jazyce Modelica

Z modelů s těmito rovnicemi byly vytvořeny FMU soubory, které byly použity místo modelů v Simulinku (viz obrázek 66).



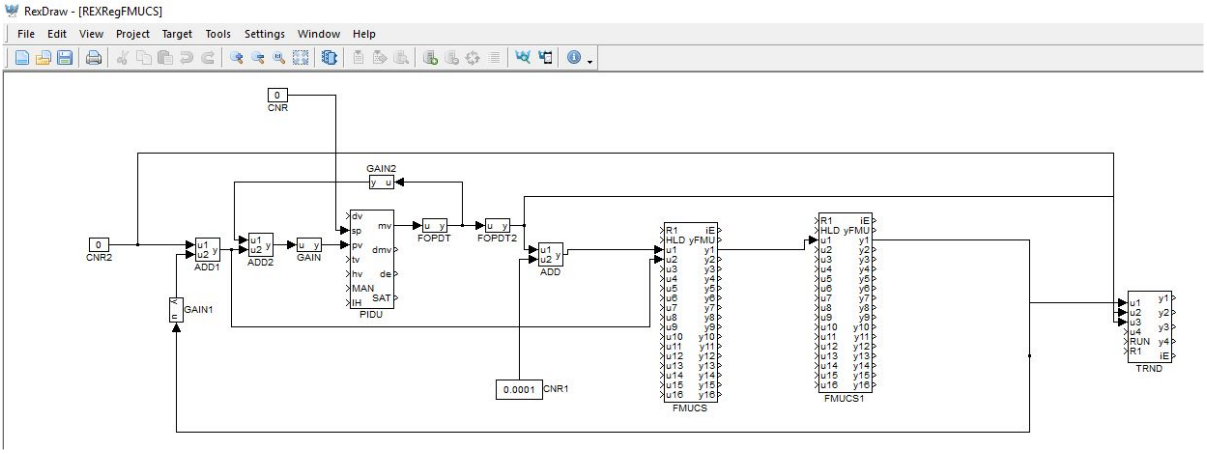

Obrázek 66: Model s FMU Model-Exchange bloky

Poté byl regulátor nahrazen regulátorem z knihovny RexLib spolu s ostatními simulinkovými bloky (viz obrázek 67).



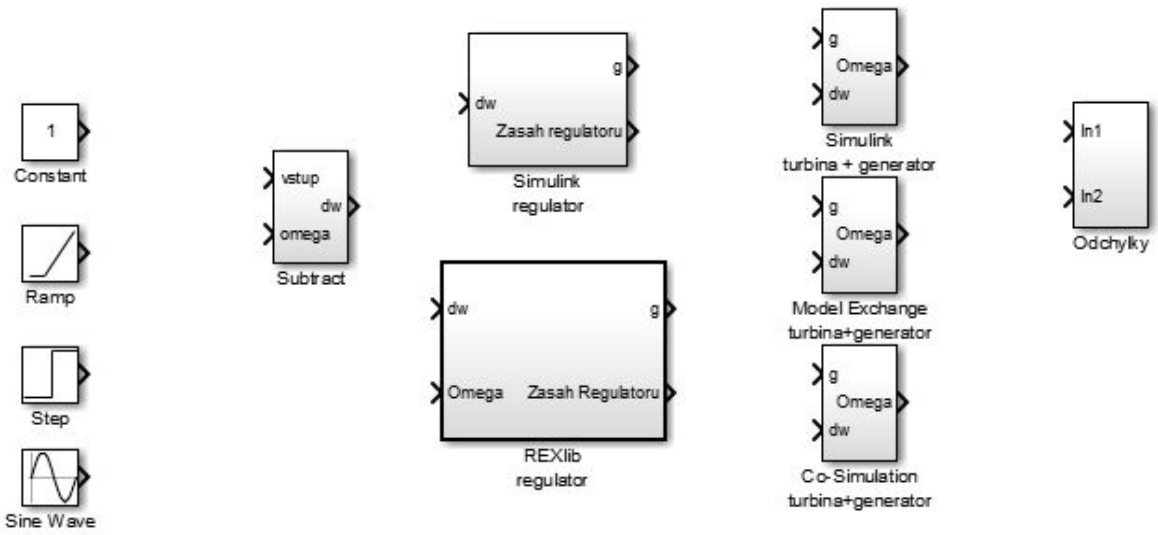

Obrázek 67: Nahrazení základních Simulink bloků bloky z RexLib

A celý model byl spuštěn v systému REX (viz obrázek 68). Tento model bohužel v některých případech nefungoval správně, a tak je v části porovnání rozdělen na 2 části: regulátor se systémy prvního řádu v jedné části a oba FMU bloky v druhé.




Obrázek 68: Model v REXu

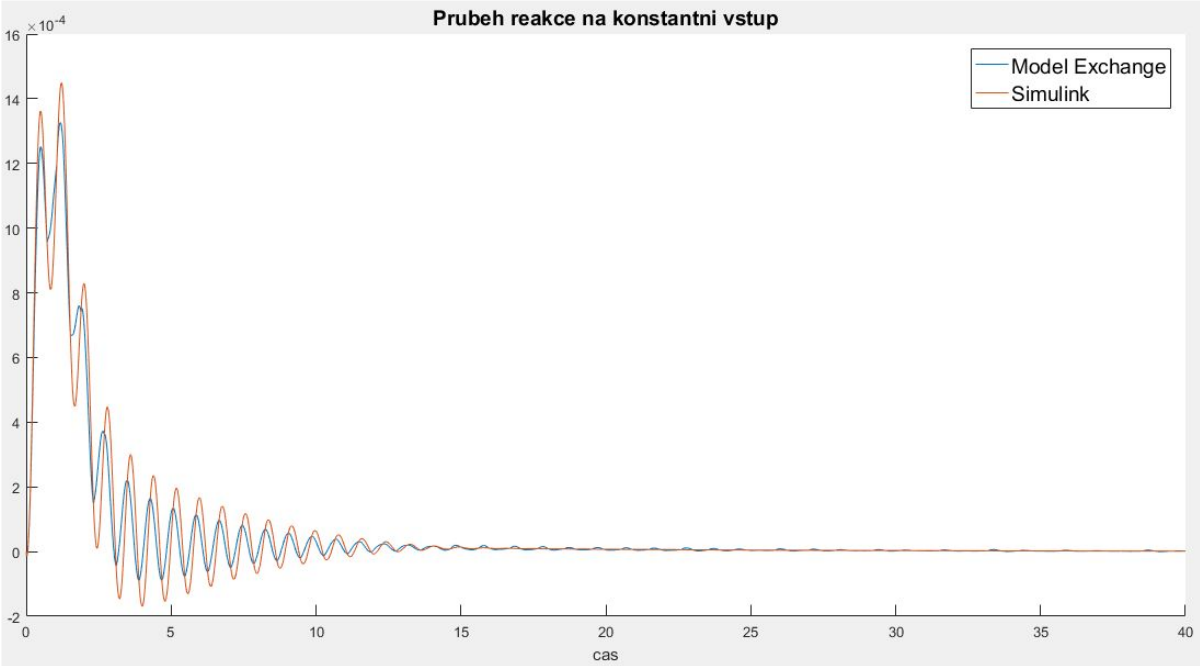
Pro porovnání byl vytvořen soubor v Simulinku obsahující všechny vytvořené modely (viz obrázek 69). Subsystémy jsou zjednodušené tak, aby bylo zapojení intuitivní. Blok pro určení odchylek je stejný jako blok u RLC modelu.



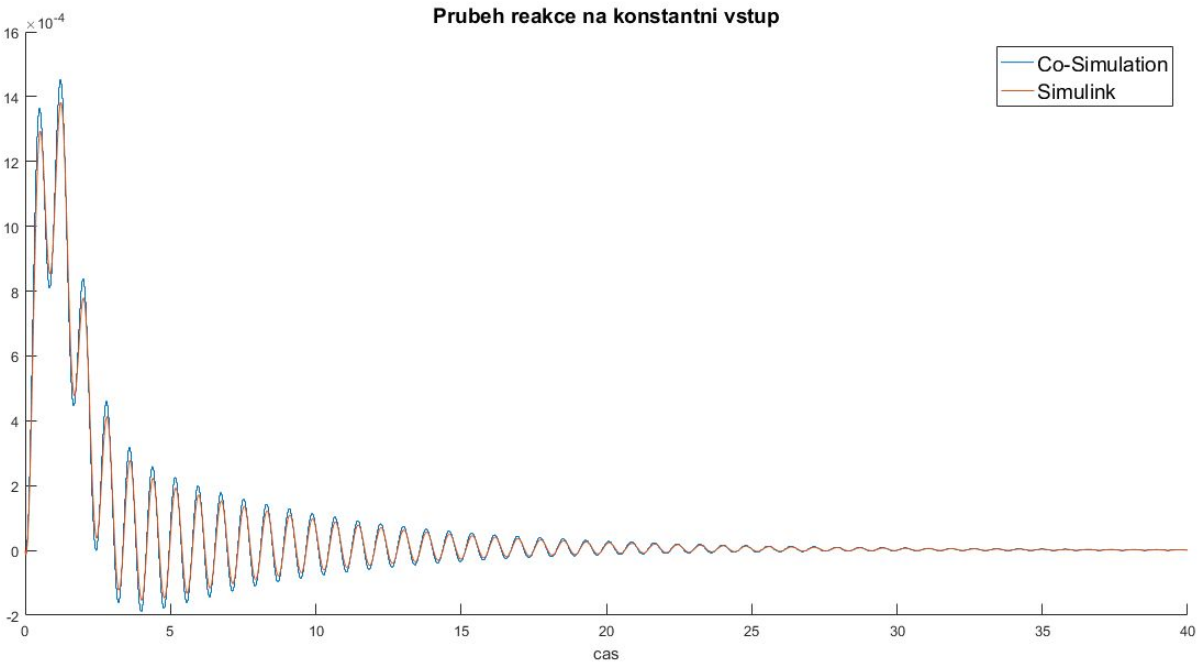

Obrázek 69: Modely turbíny a generátoru

# Porovnání modelů

Došlo k porovnání simulinkového a FMU modelu. FMU Model Exchange měl při simulaci (obrázek 70) nejmenší maximální relativní odchylku (1.2493e+02), protože se spolu se simulinkovým modelem ustálil na nule rychleji než Co-Simulation model (obrázek 71) s odchylkou 9.0431e+06.

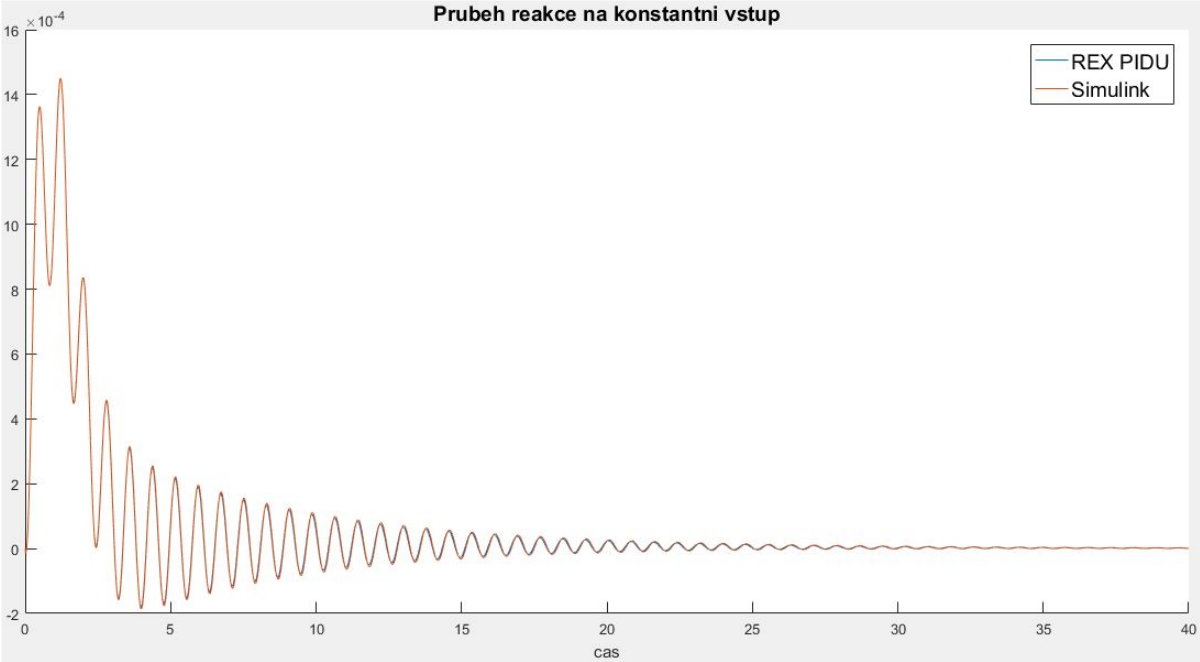



Obrázek 70: Porovnání modelu v Simulinku s Model Exchange modelem

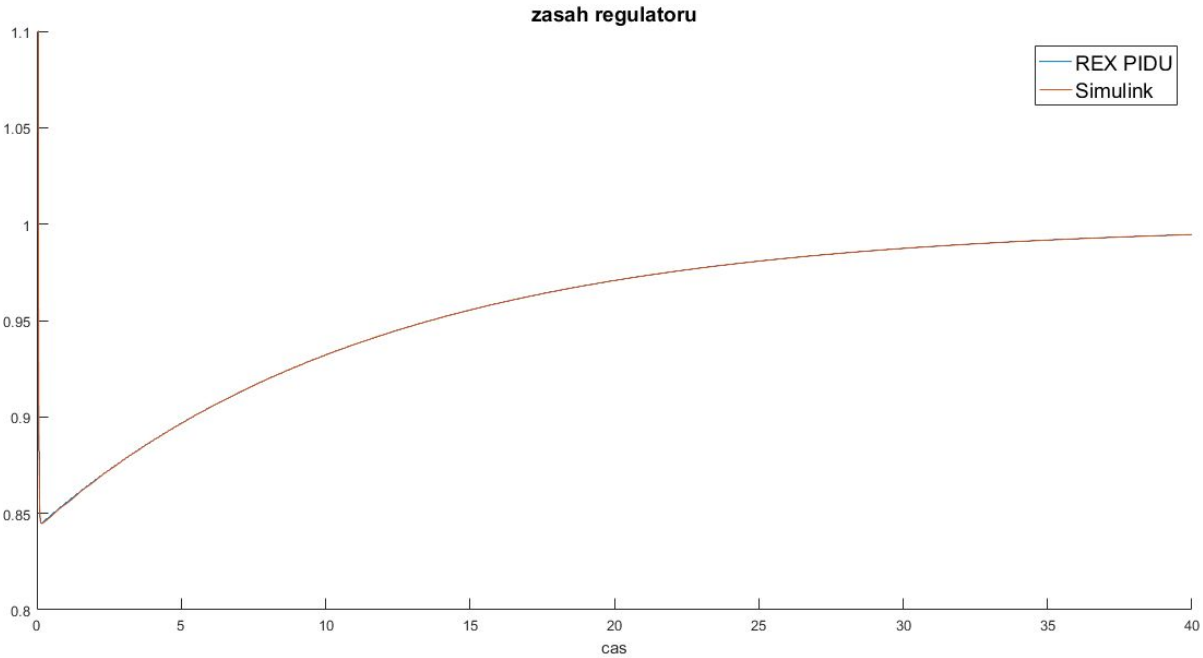



Obrázek 71: Porovnání modelu v Simulinku s Co-Simulation modelem

Došlo i k porovnání jednoduchého PI regulátoru a stejně nastaveného PIDU regulátoru z RexLibu. V grafech jsou průběhy regulované veličiny (obrázek 72) i akčních zásahů regulátorů (obrázek 73).

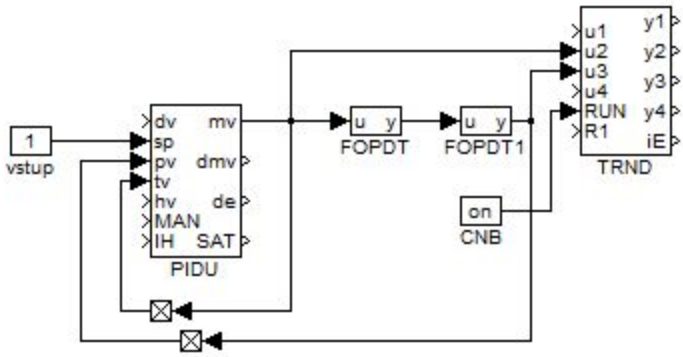



Obrázek 72: Porovnání regulátoru v Simulinku s REX PIDU regulátorem

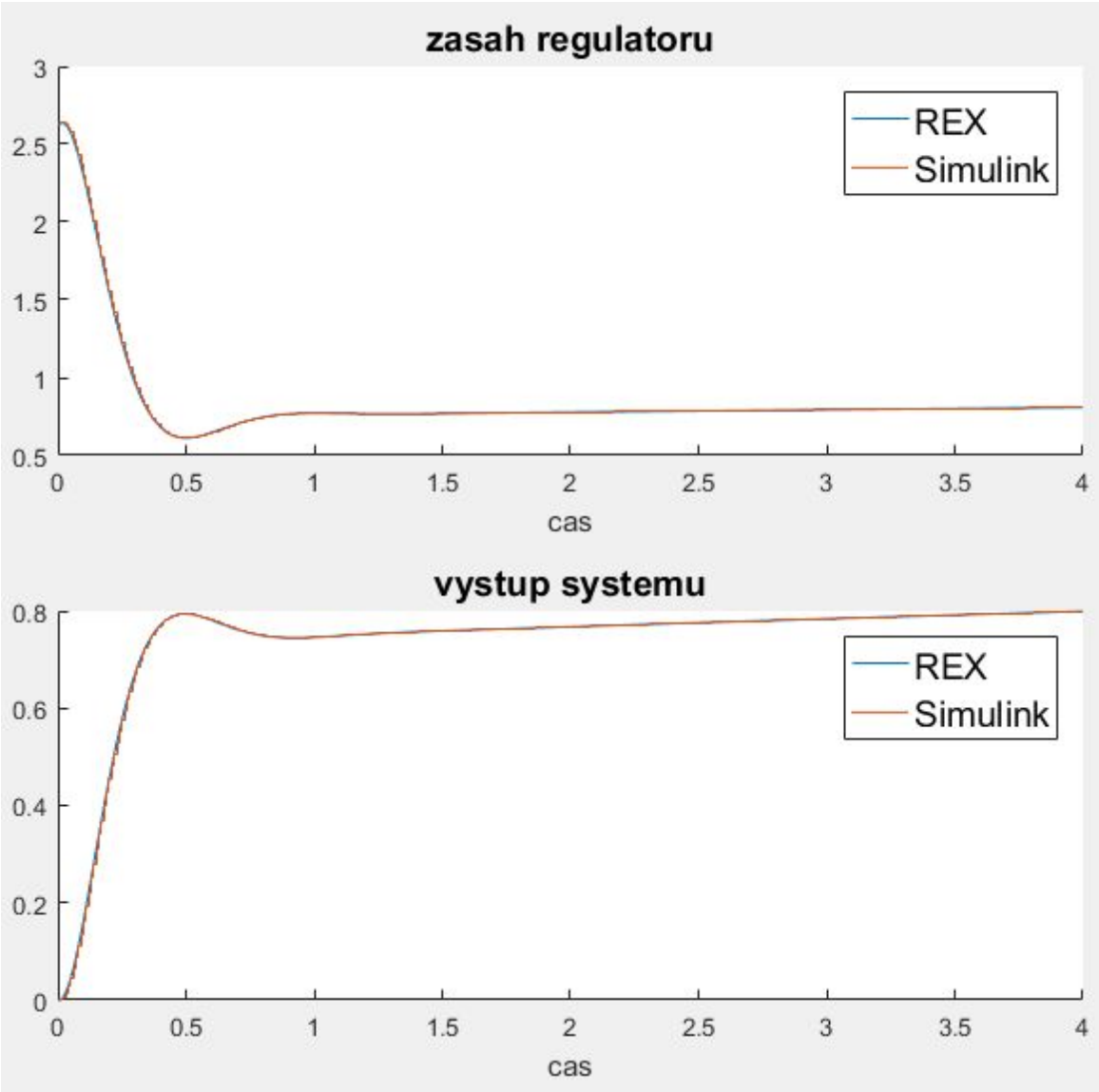



Obrázek 73: Porovnání akčních zásahů regulátorů

Regulátor se systémy prvního řádu (viz obrázek 74) byl spuštěn v systému REX a byl porovnán oproti modelu regulátoru z knihovny RexLib v Simulinku (viz obrázek 75).

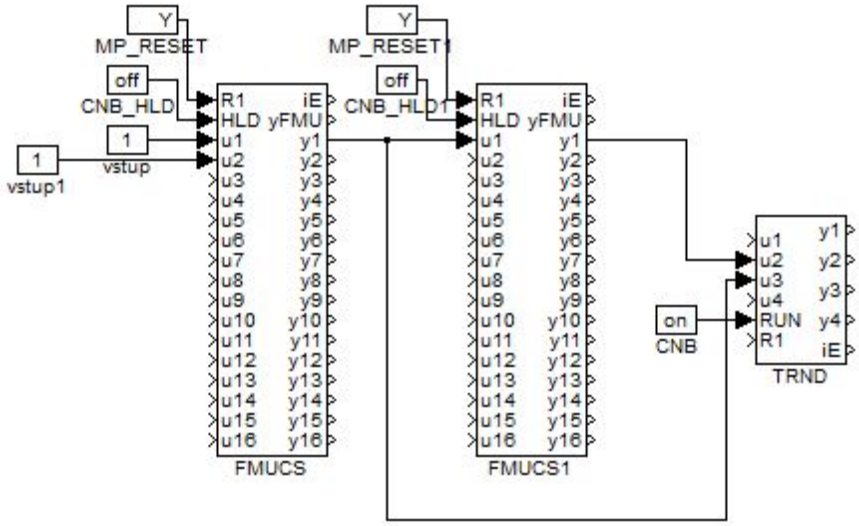



Obrázek 74: Regulátor v REXu

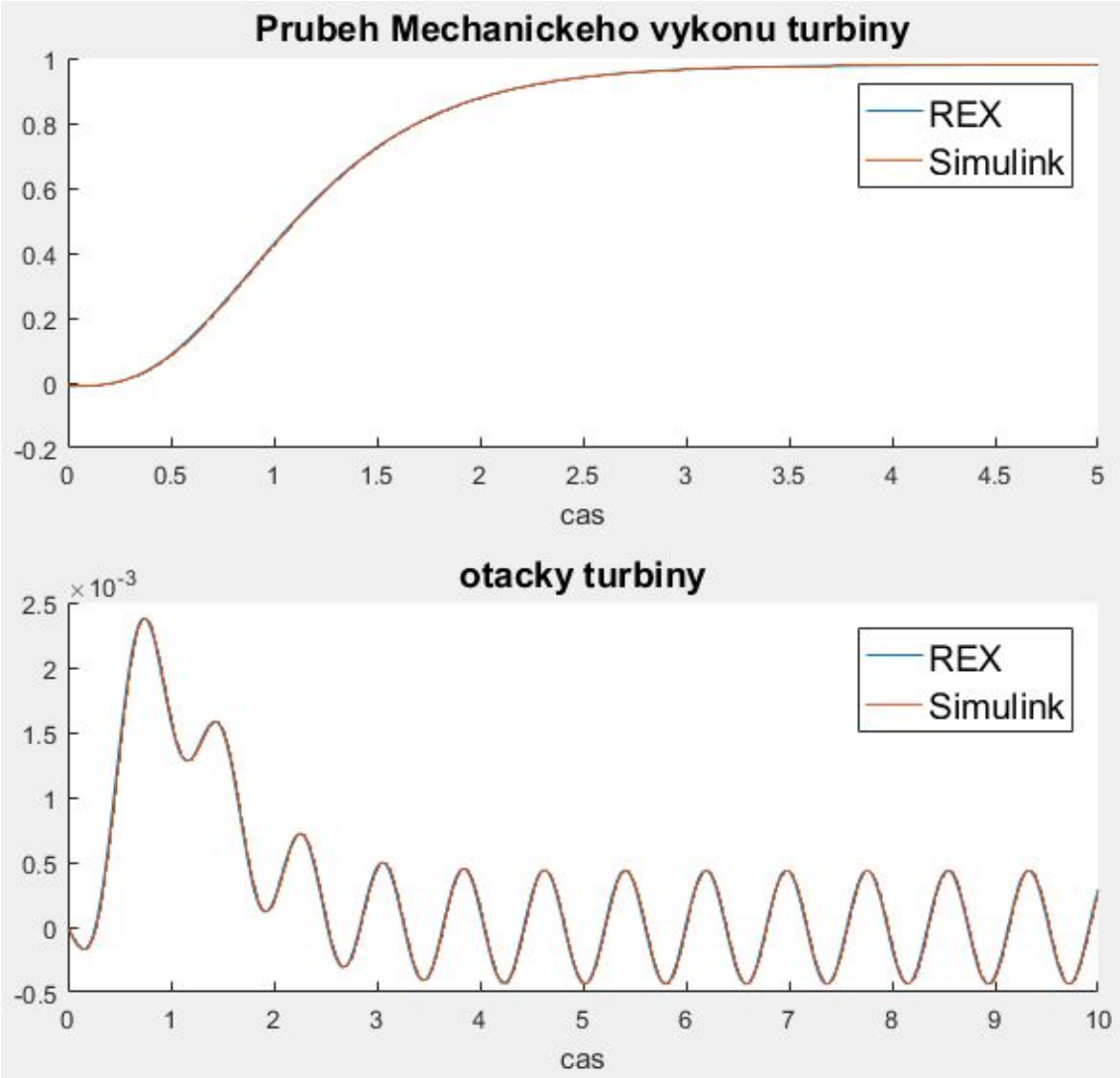



Obrázek 75: Porovnání modelů v prostředí REX a Simulink

Pro FMU modely proběhla obdobná simulace (obrázek 76). Její výsledky opět odpovídají těm v Simulinku (obrázek 77).




Obrázek 76: Bloky pro simulování FMU souborů v REXu




Obrázek 77: Porovnání chování modelů FMU Co-Simulation v REXu a Simulinku

# Závěr

Byla provedena analýza a ověření funkčnosti standardu FMI v prostředích OpenModelica a MATLAB/Simulink, a to na modelu jednoduchého fyzikálního modelu RLC obvodu popsatelného diferenciální rovnicí 2. řádu. Byly vytvořeny modely řešící diferenciální rovnici pro obvod a blokové modely s fyzikálními komponenty, všechny v Simulinku i OpenModelice, ze které byly exportovány do FMU standardu a otestovány v Simulinku. Pro porovnání jsou součástí této práce i výsledky simulací z OpenModelicy v .csv formátu. V závěru byly vytvořeny fyzikální bloky s užitím operačních zesilovačů nahrazující základní simulinkové bloky (například integrator, gain, add) a z těchto bloků byl sestaven model diferenciální rovnice pro RLC obvod, opět v Simulinku i OpenModelice. Výsledky simulací "FMI for Model Exchange" mají většinou stejný průběh (prakticky bez odchylek) jako jejich protějšky vytvořené v Simulinku.

Poté byl vytvořen návod na postavení jednoduchého modelu v OpenModelice, export tohoto modelu do FMU standardu a porovnání výsledků s ekvivalentním modelem v Simulinku.

V poslední části byl vytvořen matematický model vodní turbíny a odsimulován v Simulinku a OpenModelice, odkud byl převeden do FMU standardu. Byl navržen jednoduchý PI regulátor pro otáčky turbíny a s obdobou tohoto regulátoru byl vytvořen model v systému REX, ten se bohužel celý odsimulovat nepovedlo, ale jeho část se podařilo porovnat se stejnou částí v Simulinku.

Během tvorby této práce jsem získal širší povědomí o funkčnosti FMU standardu, spolu s množstvím problémů, které nastávají při přesunu modelu z jednoho prostředí do jiného. Příčinu velké části těchto problémů se podařilo objasnit a vyřešit.

# Zdroje

[1] V. Kučera, P. Zídek, P. Balda, M. Čech, J. Königsmarková, J. Reitinger: Zpráva o testování RT simulátoru pro komponentové modely

[2] Popis FMI/FMU na wikipedii:
https://en.wikipedia.org/wiki/Functional_Mock-up_Interface

[3] FMI/FMU - oficiální stránky: https://www.fmi-standard.org/

[4] Torsten Blochwitz, Martin Otter: The Functional Mockup Interface for Tool independent Exchange of Simulation Models (dostupné v příloze)

[5] MATLAB - oficiální stránky: https://www.mathworks.com/products/matlab.html

[6] Martin Otter: Modelica Overview
dostupné na:
https://www.modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf

[7] OpenModelica - oficiální stránky: https://openmodelica.org/

[8] REX Controls s.r.o.: Uživatelská příručka vývojového nástroje RexDraw
dostupné na:
https://www.rexcontrols.cz/media/2.50.1/doc/PDF/CZECH/RexDraw_CZ.pdf

[9] obrázek RLC obvodu
https://www.mathworks.com/help/simulink/ug/seriesrlc.png

[10] Milan Vosáhlo: Projekt 4 (Modely RLC obvodu v SimScape a OpenModelice)

[11] Karel Kubíček, Milan Vosáhlo: Projekt 5

[12] Operační zesilovače na wikipedii: https://cs.wikipedia.org/wiki/Zapojen%C3%AD_s_opera%C4%8Dn%C3%ADm_zesilova%C4%8Dem

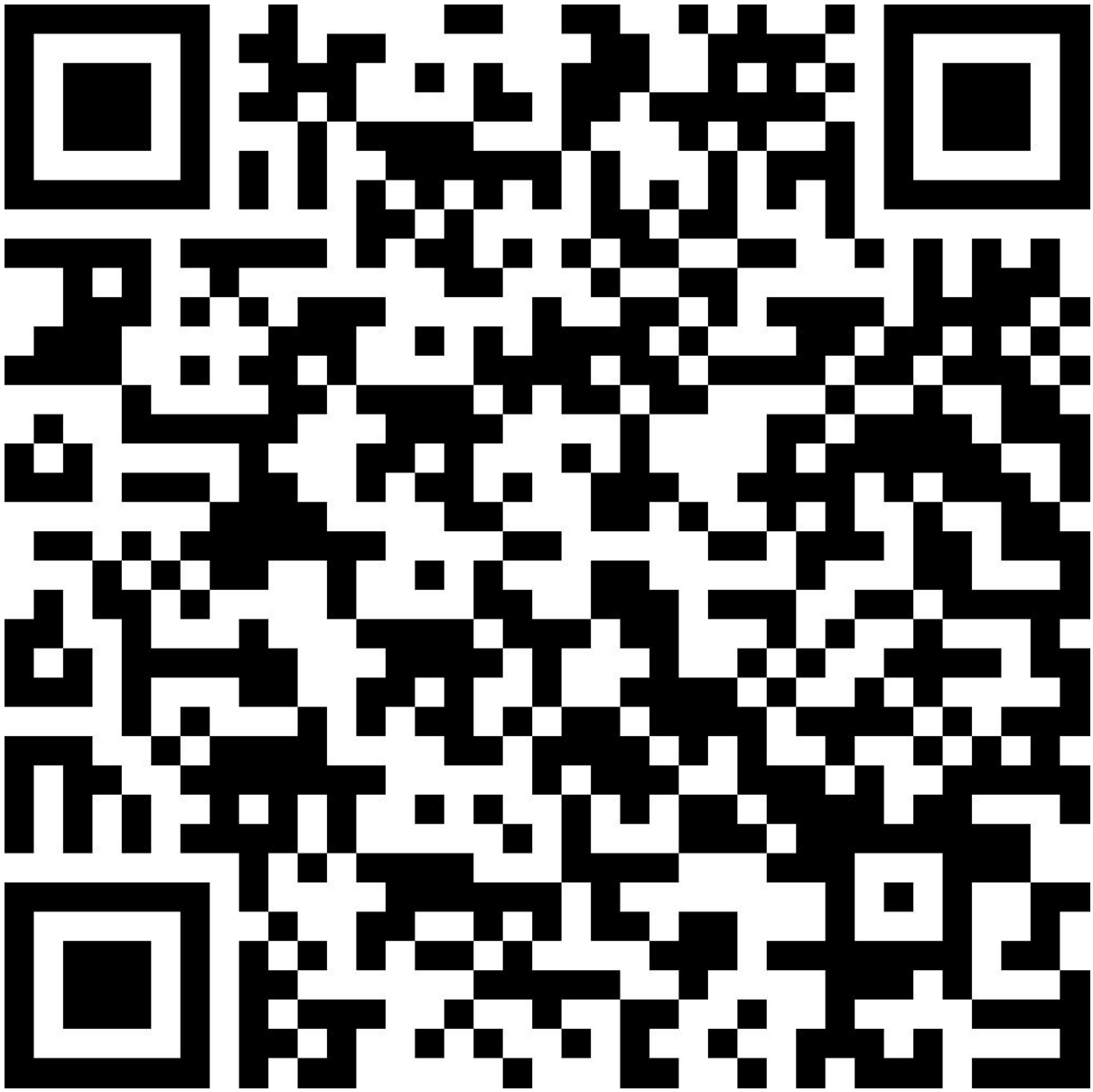
[13] E. De Jaeger, N. Janssens, B. Malfliet, F. Van De Meulebroeke: Hydro turbine model for system dynamic studies

[14] Karel Kubíček, bakalářská práce, 2017

# Příloha

Modely RLC obvodu, Projekt 4 a 5, Bakalářská práce kolegy Kubíčka a další přílohy jsou k dispozici ve složce na Google drive pod následujícím odkazem:

https://drive.google.com/drive/folders/0B1kKkDuf2hhjLXBWd2lBekx4RnM?usp=sharing



V případě nefunkčnosti odkazu nebo dotazů mě můžete kontaktovat na e-mailové adrese: milanvosahlo@mail.com