VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VYUŽITÍ OPTIMALIZAČNÍCH METOD PRO ODHAD PARAMETRŮ SIMULAČNÍCH MODELŮ

APPLICATION OF OPTIMIZATION METHODS FOR PARAMETERS ESTIMATION OF SIMULATION MODELS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Appel**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Robert Grepl, Ph.D.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Martin Appel**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Robert Grepl, Ph.D.**
Akademický rok:	2015/16

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Využití optimalizačních metod pro odhad parametrů simulačních modelů

Stručná charakteristika problematiky úkolu:

Při modelování a řízení dynamických systémů je podstatná znalost jejich parametrů. Některé tyto parametry známe nebo je lze jednoduše změřit, řadu parametrů však přímo měřit nelze (např. součititele tlumení a tření, tepelné odpory a další). Tyto parametry musíme odhadovat na základě porovnání experimentálně naměřených dat a výsledků simulací.

Tato práce se bude zabývat aplikací různých optimalizačních metod na problém odhadu parametrů simulačních modelů. V prostředí MATLAB/Simulink existuje nástroj Simulink Parameter Estimation, který má však některá výrazná uživatelská omezení. Cílem práce je navrhnout a implementovat software s výrazně lepšími vlastnostmi a možnostmi. Součástí práce bude testování s využitím dat naměřených na reálných laboratorních modelech.

Cíle diplomové práce:

1) Provést kritické zhodnocení stávajícího stavu, především nástroje SPE. Navrhnout klíčová vylepšení.
2) Navrhnout a implementovat strukturu ukládání a práce s modely, daty, parametry a výsledky simulací.
3) Navrhnout a implementovat funkce pro provádění odhadu s využitím metod Optimization toolboxu a vlastních metod (např. Grid, Monte Carlo, Genetický algoritmus). Navrhnout a implementovat funkce pro uživatelsky přívětivou a užitečnou vizualizaci a analýzu výsledků simulačních experimentů.
4) Vypracovat několik ukázkových studií odhadu parametrů na simulačních i reálných modelech.

Seznam literatury:

Nelles, O: Nonlinear System Identification, Springer 2001
Jung, L.: System Identification, 2009

Valášek, M.: Mechatronika, Vydavatelství ČVUT 1995

Noskievič, P.: Modelování a identifikace systémů, 1999

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato práce se zabývá problematikou hledání odhadu parametrů. V rámci práce byl vytvořen nový program, který umožňuje provádět hledání odhadu parametrů, podobně jak nástroj Parametr estimation, který je součástí rozšiřujícího balíčku Matlabu. Nově vytvořený program vhodnou vizualizací a novými funkcemi umožňuje pro některé případy nalezení lepšího odhadu parametrů než program od Mathworks.

Summary

This thesis deals with parameters estimation search problematics. Newly-made software is proposed within a frame of this work and it replaces and supplements the Parameter Estimation tool, which is a part of Matlab toolbox. New software proposes suitable visualisation and new functions, which may lead to better solutions than build-in tools in Matlab.

Klíčová slova

Hledaní odhadu parametrů, Parametr estimation, Simulace, Matlab, Simulink, Genetický algoritmus, Monte Carlo, Grid.

Keywords

Parameter estimation search, Parametr estimation, Simulation, Matlab, Simulink, Genetic algorithm, Monte Carlo, Grid.

Bibliografická Citace

APPEL, M. *Využití optimalizačních metod pro odhad parametrů simulačních modelů*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2016. 100 s., Vedoucí diplomové práce: doc. Ing. Robert Grepl, Ph.D.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a že jsem uvedl veškeré použité zdroje a literaturu.

Martin Appel

Brno . . . .

Rád bych poděkoval vedoucímu diplomové práce doc. Ing. Robertu Greplovi, Ph.D. za cenné rady a připomínky. Též bych chtěl poděkovat mechatronické laboratoři za poskytnuté zázemí a možnosti pracovat na zajímavých projektech. Jako poděkování za podporu jsem se rozhodl program, vytvořený v rámci této práce, pojmenovat "Mechlab's parametr estimation."

Martin Appel

Obsah

1 Úvod

Pro lepší řízení soustavy než obyčejným PID regulátorem je znalost soustavy nezbytná. Pokud je možné soustavu matematicky popsat, lze také vytvořit model, který reprezentuje do určité míry soustavu. Tento model obsahuje řadu parametrů, které je možné v lepším případě změřit, v opačném případě je potřeba je získat odhadem pomocí naměřených dat.

Myšlenka je jednoduchá. Postupnou úpravou parametrů systému se mění odezva na vstupní signál, který je porovnáván s naměřeným signálem na soustavě. Ovšem prohledávací prostor bývá obvykle obrovský a skrývá mnoho lokálních minim. Nelze tedy použít jenom některou z gradientních metod ani nelze prohledat celý prostor.

„Identifikace systému je umění a věda, která se zabývá vytvořením matematického modelu dynamického systému z naměřených vstupních a výstupních dat“[14]. Umění, tedy znalosti a schopnosti uživatele, jsou obvykle nezastupitelnou součástí. Proto je potřeba umožnit uživateli, aby byl součástí procesu hledání.

1.1 Práce s Parametr estimation toolbox

Program PE (parametr estimation toolbox) je mocný nástroj, který umožňuje nalezení odhadu parametrů, pokud máme model a naměřená data. Ovšem má několik nedostatků, popsaných v kapitole 2.5. Jedná se hlavně o filozofii, že je model, nad kterým běží program PE. Jedná se tedy o práci jen s jediným modelem. Program je pro uživatele black-box, kde neví, jak uvnitř pracuje a ani se nedostane k průběhu hledaní parametrů.

S narůstajícím počtem hodin strávených nad programem PE se vkrádá myšlenka, jestli to, jak je program navržený, je ideální varianta a jestli nejde přijít s něčím lepším. Tedy vzniká motivace vytvořit nástroj, který poskytne nový přístup k problematice.

1.2 Stanovené cíle

Nový program, který bude vytvořen v rámci této práce, by měl poskytnout uživateli možnost zohlednit jeho znalost soustavy tak, aby došlo k synergické integraci schopnosti technicky zdatného uživatele a výpočetní síly počítače. Jedná se tedy o inženýrský nástroj, určený pro ty, co to s hledáním odhadu parametrů myslí vážně. Nejde tedy o program, který během lusknutí prstu nalezne parametry a napíše, že hledání dopadlo úspěšně. Inženýr musí být skeptický k výsledkům z programů, u kterých neví, jak vevnitř fungují.

Program by tedy měl být schopen poskytnout veškeré informace o průběhu hledání odhadu parametrů, také by měl nabízet možnost konkrétně nastavit prohledávací metodu nebo ji dokonce vytvořit.

Hlavním přínosem nového programu by měla být schopnost pracovat s více modely, což by umožnilo vytvářet různé experimenty, které sdílejí parametry. Nebo porovnat několik modelů, jak reprezentují soustavu. Popis požadavků na nově vytvořený program je v kapitole 3.

2 Rešerše

Tato kapitola se bude zabývat nástrojem PE (Parameter Estimation), přičemž primárně bude zaměřena na verzi nástroje, dostupnou ve verzi MATLAB R2013b. Primárním zdrojem pro tuto kapitolu je nápověda MATLABu [2] a získaná zkušenost s nástrojem.

2.1 Nástroj Parameter Estimation

Nástroj PE je součástí doplňkového balíčku *Simulink Design Optimization*. Nástroj odhaduje parametry a počáteční stavy modelu se známou strukturou v Simulinku pomocí naměřených dat. Tento nástroj zvyšuje přesnost modelu tak, aby model reprezentoval naměřené chování hardwaru (na měřené soustavě). Pro použití tohoto nástroje je nutné mít připravený model v Simulinku s nějakými vstupy, výstupy a parametry. Tento model je chápán jako grey-box. Pomocí optimalizačních algoritmů následně nástroj hledá takové parametry, které minimalizují stanovenou kriteriální funkci (rozdíl mezi naměřenou a simulovanou odezvou). Lze tedy například automaticky odhadnout odpor motoru, indukčnost a moment setrvačnosti z naměřených dat napětí a rychlosti motoru. S nástrojem lze pracovat buď prostřednictvím grafického uživatelského rozhraní, nebo pomocí příkazové řádky.

Pomocí tohoto nástroje lze:

- Importovat a předzpracovat naměřená data.
- Nalézt nejvýznamnější odhadované parametry (pomocí nástroje Sensitivity Analysis – dostupné až od verze R2016a).
- Odhadovat parametry a počáteční stavy modelu a monitorovat postup estimace (odhadu).
- Validovat (verifikovat) výsledky estimace.

Nástroj umožňuje odhad a validaci více parametrů modelu zároveň a je možné zavést meze možných hodnot parametrů.

Z nástroje lze generovat i kód v MATLABu a urychlit odhad parametrů pomocí paralelních výpočtů a funkce *Simulink fast restart* (dostupné až od verze R2015b).

2.2 Formulace odhadu parametrů

2.2.1 Přehled odhadu parametrů jako optimalizační úlohy

Tento software formuluje odhad parametrů jako optimalizační úlohu. Řešením této optimalizační úlohy jsou odhadované hodnoty parametrů. Tato optimalizační úloha se skládá z:

- x - *Navrhové proměnné*. Parametry modelu a počáteční stavy, které mají být odhadovány.

– $F(x)$ – *Účelová funkce.* Funkce, která počítá míru rozdílu mezi naměřenou a simulovanou odezvou. Tuto funkci lze také nazvat jako *hodnotící funkce*, *kriteriální funkce* nebo *chyba estimace (odhadu).*

– (Volitelné) - *Meze.* Omezení hodnot odhadovaných parametrů.

– (Volitelné) $C(x)$ – *Omezovací funkce.* Funkce, která specifikuje omezení návrhových proměnných.

Optimalizační řešič ladí hodnoty návrhových proměnných s cílem splnit stanovené cíle a omezení. Přesná formulace optimalizace závisí na použité optimalizační metodě.

2.2.2 Kriteriální funkce

Software ladí parametry modelu za účelem dosažení simulované odezvy y_sim, která sleduje naměřenou odezvu nebo referenční signál y_ref. Aby toho bylo dosaženo, řešič minimalizuje kriteriální funkci, neboli odchylku estimace, tedy míru rozdílu mezi naměřenou a simulovanou odezvou. Kriteriální funkce $F(x)$ je účelovou funkcí optimalizační úlohy.

Hrubá odchylka odhadu e(t) je definovaná jako:

$$e(t) = y_ref(t) - y_sim \tag{2.1}$$

$e(t)$ je také označováno jako *reziduály odchylek* nebo jednoduše *reziduály.*

Software vyhodnocuje kriteriální funkci pro určitý časový interval. Tento interval je závislý na *časové bázi naměřeného signálu* a *časové bázi simulovaného signálu.*

– *Časová báze naměřeného signálu* se skládá ze všech časových bodů, pro které je naměřený signál specifikovaný. V případě více naměřených signálů je časová báze sjednocením časových bodů všech naměřených signálů.

– *Časová báze simulovaného signálu* se skládá ze všech časových bodů, pro které je model simulován.

V případě, že model používá řešič s proměnnou délkou kroku, pak se časová báze simulovaného signálu může mezi jednotlivými optimalizačními iteracemi měnit. Časové báze simulovaného a naměřeného signálu se mohou lišit. Software vyhodnocuje kriteriální funkci pouze pro časový interval, který je společný pro obě báze. Ve výchozím nastavení software používá ve společném časovém intervalu pouze časové body specifikované naměřeným signálem.

– V GUI lze stanovit časy začátku a konce simulace v oblasti *Simulation time* v dialogovém okně *Simulation options.*

– V příkazovém řádku software stanoví čas konce simulace jako poslední bod časové báze naměřeného signálu.

Software Simulink Design Optimization poskytuje následující kriteriální funkce pro zpracování $e(t)$

Součet čtverců odchylek SSE

Součet čtverců odchylek je výchozí metoda v programu PE. V programu je pod názvem *SSE.*

$$SSE = \sum_{i=1}^{n} e(t)^2 \tag{2.2}$$

Součet absolutních odchylek SAE

Součet absolutních odchylek dává stejnou váhu na všechny chyby $e(t)$. V programu je pod názvem *SAE*.

$$SAE = \sum_{i=1}^{n} |e(t)| \tag{2.3}$$

Hrubá odchylka - Residuals

Tato volba je dostupná, pouze pokud se s programem pracuje pomocí příkazů. V programu je pod názvem *Residuals*.

$$F(x) = \begin{bmatrix} e(0) \\ \vdots \\ e(N) \end{bmatrix} \tag{2.4}$$

Uživatelem definovaná funkce

Uživatel má možnost definovat svoji metodiku hodnocení. Tuto volbu lze použít jen při práci s PE pomocí příkazů.

2.2.3 Meze a vazby

Na základě znalosti systému lze stanovit meze pro navrhované proměnné (odhadované parametry modelu). Meze jsou vyjádřeny jako:

$$\underline{x} \leq x \leq \bar{x} \tag{2.5}$$

Kde: $\underline{x}$ je spodní a $\bar{x}$ horní mez navrhované proměnné

Například u oscilátoru musí být odhadovaná hmotnost větší než nula a menší než nekonečno. Tyto meze jsou vyjádřeny jako:

$$0 < x < \infty \tag{2.6}$$

Při práci s PE pomocí příkazů lze také stanovit další vazby $C(x)$ pro navrhované proměnné. $C(x)$ může být lineární nebo nelineární a může popisovat rovnosti či nerovnosti. $C(x)$ může také specifikovat vazby mezi více parametry. Například v jednoduchém modelu tření může $C(x)$ předepsat, že statický koeficient tření x_2 musí být větší nebo roven dynamickému koeficientu tření x_1. Jedním ze způsobů vyjádření této vazby je:

$$C(x) = x_1 - x_2 \tag{2.7}$$

$$C(x) \leq 0 \tag{2.8}$$

2.2.4 Optimalizační metody

Nonlinear Least Squares

Při práci s PE pomocí skriptu se metoda jmenuje `lsqnonlin`. Minimalizuje čtverce reziduálů, je to doporučená metoda pro hledání odhadu parametrů. Tato metoda vyžaduje vektor reziduálů odchylek, vypočtený za použití fixní časové báze. Tento přístup není vhodný v případě skalární kriteriální funkce, nebo pokud se počet reziduálů odchylek může mezi iteracemi měnit. Tato metoda používá funkci *lsqnonlin*.

lsqnonlin řeší úlohy nelineárních nejmenších čtverců, včetně úloh typu *curve-fitting* (prokládání křivek).

Gradient Descent

Při práci s PE pomocí skriptu se metoda jmenuje `fmincon`. Obecný nelineární řešič používá gradient kriteriální funkce. Tento přístup je vhodné použít, pokud je žádoucí stanovit jednu z následujících možností, či jejich libovolnou kombinaci:

- Uživatelem definovaná kriteriální funkce
- Parametrové vazby
- Signálové vazby

Tato metoda používá funkci *fmincon*.

fmincon hledá minimum omezené nelineární skalární funkce více proměnných. Obecně se to označuje jako *constrained nonlinear optimization* (omezená nelineární optimalizace) nebo *nonlinear programming* (nelineární programování).

Simplex Search

Při práci s PE pomocí skriptu se metoda jmenuje `fminsearch`. Založeno na algoritmu *Nelder-Mead*. Tento přístup nepoužívá gradient kriteriální funkce. Použijte tento přístup, pokud kriteriální funkce nebo vazby nejsou spojité nebo diferencovatelné. Tato metoda používá funkce *fminsearch* a *fminbnd*. *fminbnd* se používá, je-li optimalizován jeden skalární parametr. V opačném případě se použije *fminsearch*. S *fminsearch* nelze specifikovat meze parametrů $\underline{\mathrm{x}} \leq x \leq \bar{x}$.

fminsearch hledá minimum neomezené skalární funkce více proměnných. Obecně se to označuje jako *unconstrained nonlinear optimization* (neomezená nelineární optimalizace).

fminbnd hledá minimum funkce jedné proměnné ve fixním invervalu.

Pattern Search

Při práci s PE pomocí skriptu se metoda jmenuje `patternsearch`. Je to přímá metoda vyhledávání, založená na zobecněném algoritmu *pattern search* (hledání vzorů). Tato metoda nepoužívá gradient kriteriální funkce. Je vhodné použít tento přístup, pokud kriteriální funkce nebo vazby nejsou spojité nebo diferencovatelné. Tato metoda používá funkci *patternsearch*.

patternsearch hledá minimum funkce pomocí *pattern search* (hledání vzorů).

2.3 Příprava dat

2.3.1 Podporovaná data

K hledání odhadu parametrů a počátečních podmínek Simulink modelu s jedním či více vstupy a výstupy jsou zapotřebí naměřená signálová data. Při měření dat je nutné zvolit vhodné buzení systému, aby systém nebyl v rovnovážném stavu a bylo možné zachytit dynamiku systému. Vhodným buzením může být například skok, impulsní vstupy, šum nebo libovolný jiný vhodný signál, který „vybudí v systému zrychlení“. PE umožňuje hledat odhad parametrů modelu z následujících typů dat:

- *Time-domain data* data, která mají jednu nebo více vstupních proměnných $u(t)$ a jednu nebo více výstupních proměnných $y(t)$, vzorkované jako funkce času.

- *Time-series data* data uložená v objektech typu „time-series“.

PE hledá odhad parametrů modelu porovnáním naměřených signálových dat se simulačními daty získanými ze Simulink modelu. Software hledá odhad parametrů a počátečních podmínek stavů pomocí optimalizačních metod minimalizací uživatelem zvolené kriteriální funkce. Kriteriální funkce typicky počítá odchylku mezi naměřenými a simulovanými daty pomocí metody nejmenších čtverců.

2.3.2 Požadavky na model pro import dat

Před analýzou a předzpracováním naměřených dat pro hledání odhadu (estimačních dat) je nutné přiřadit data k jednotlivým kanálům modelu. Aby bylo možné přiřadit data, musí model v Simulinku obsahovat jeden z těchto prvků:

- Blok *Inport* v nejvyšší úrovni modelu. Pokud model již obsahuje pevný vstupní blok (např. *Step*), pak blok *Inport* není potřeba.

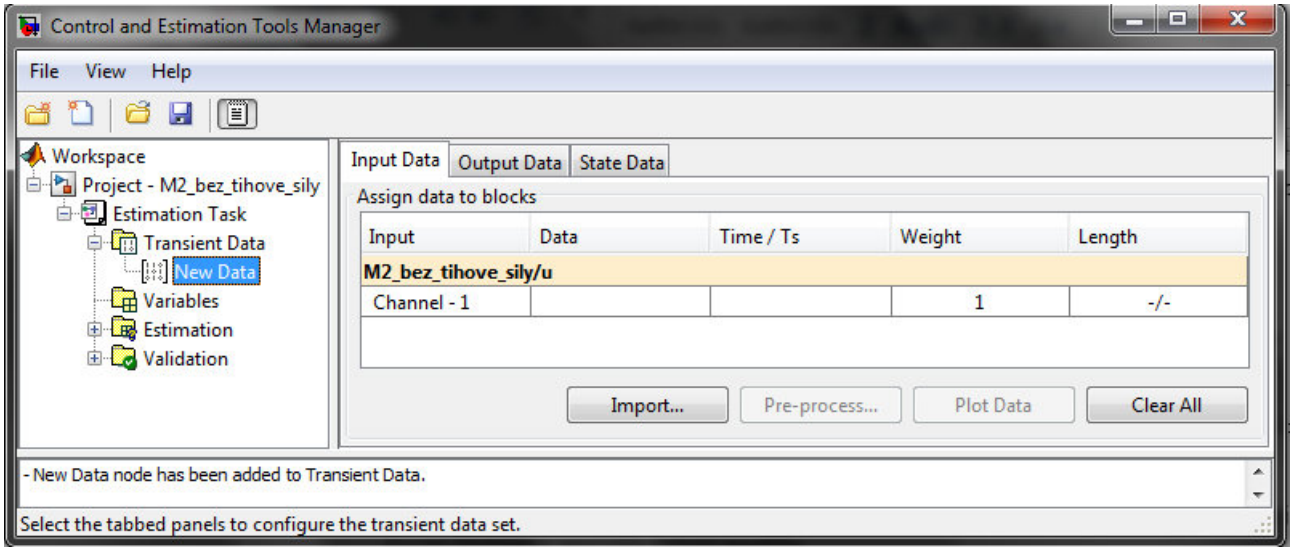
- Blok *Outport* v nejvyšší úrovni modelu.

- Zaznamenaný (*logged*) signál. Zaznamenaný signál může být signál v nejvyšší úrovni modelu nebo signál v subsystému modelu.

Pro zaznamenání signálu je potřeba v Simulink editoru vybrat daný signál, kliknout na šipku u tlačítka ***Record*** a kliknout na ***Log/Unlog Selected Signals***.

V GUI *Control and Estimation Tools Manager*, řádky na kartě ***Input Data*** odpovídají *Inport* blokům v nejvyšší úrovni modelu. Viz obrázek 2.1. Podobně řádky na kartě ***Output Data*** odpovídají buď *Outport* blokům v nejvyšší úrovni nebo zaznamenaným signálům v modelu.

Přidání *Inport* nebo *Outport* bloku nebo označení signálu pro zaznamenání vytvoří nový řádek v odpovídající záložce. Nový řádek lze použít k importu estimačních dat pro odpovídající signál. Pro zobrazení nového řádku je nutné kliknout na tlačítko ***Update Task*** v uzlu ***Estimation Task*** v GUI *Control and Estimation Tools Manager*.



Obrázek 2.1: Input Data

2.3.3 Import dat

Než se začnou importovat data, musí se vytvořit a nastavit úloha hledání odhadu konfigurací příslušných parametrů, řešičů a kriteriálních funkcí. Software PE poskytuje grafické uživatelské rozhraní (GUI), které umožňuje vytvoření projektu hledání odhadu rychle a jednoduše.

Pro vytvoření projektu hledání odhadu:

- Otevřít Simulink model. Model pro účely tohoto návodu představuje mechanický oscilátor. Tento model obsahuje vstupní (Inport) blok u a výstupní (Outport) blok y pro import vstupních a výstupních dat.

- GUI se otevře *Control and Estimation Tools Manager* výběrem ***Analysis - Parameter Estimation*** v okně Simulink modelu.

Strom projektu zobrazuje název projektu *Project – M2_bez_tihove_sily*. Úlohy hledání odhadu jsou uspořádány uvnitř uzlu *Estimation Task*.

Poznámka: Pro vykonání úloh hledání odhadu musí zůstat Simulink model otevřený.

Po vytvoření projektu hledání odhadu, jak je popsáno v předchozím textu, je možné importovat estimační data do GUI.

Import naměřených dat:

- V GUI se zvolí ***Transient Data*** v uzlu ***Estimation Task***.

- Pravým tlačítkem myši se klikne na ***Transient Data*** a zvolí se ***New*** pro vytvoření uzlu ***New Data***. Případně se může pro vytvoření tohoto uzlu použít tlačítko ***New***.

- Vybere se uzel ***New Data*** v uzlu ***Transient Data***.

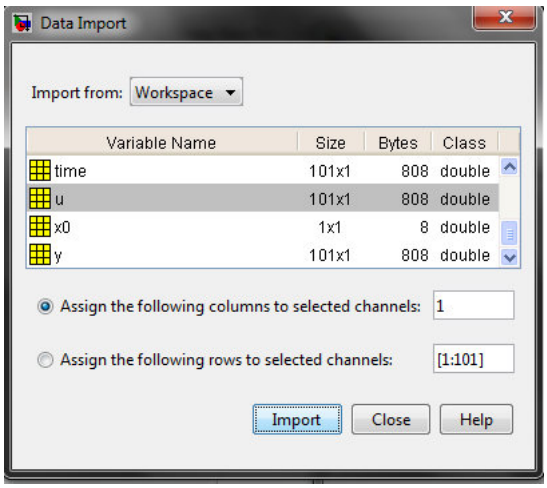
Řádky tabulky na kartě *Input Data* odpovídají vstupnímu *Inport* bloku u v modelu M2. Podobně řádky na kartě *Output* Data odpovídají výstupnímu *Outport* bloku y.

Poznámka: Aby bylo možné importovat data, musí Simulink model obsahovat blok *Inport* nebo *Outport* nebo zaznamenané (*logged*) signály. Pro více informací viz kapitola 2.3.2.

Nyní je nutné importovat vstupní a výstupní data, jak je popsáno v následujících sekcích:

Import vstupních dat a časového vektoru

- V uzlu ***New Data***se klikne na kartu ***Input Data***.
- Kliknutím pravým tlačítkem na buňku ***Data*** a zvolením ***Import*** se otevře dialogové okno *Data Import*. K otevření tohoto dialogového okna lze také použít tlačítko ***Import***.
- V dialogovém okně ***Data Import*** se vybere u ze seznamu proměnných. Viz obrázek 2.2.
- Dále tlačítko ***Import***.
- V kartě ***Input Data*** se vybere buňka ***Time/Ts***.
- Vybere se ***time*** v dialogovém okně ***Data Import***.
- Kliknutím na tlačítko ***Import*** se vybere import časového vektoru pro vstupní data.
- Kliknutím na tlačítko ***Close*** se zavře dialogové okno *Data Import*.



Obrázek 2.2: Dialogové okno Data Import

Import výstupních dat a časového vektoru

- V uzlu ***New Data*** se klikne na kartu ***Output Data***.
- Kliknutím pravým tlačítkem na buňku ***Data*** a zvolením ***Import*** se otevře dialogové okno *Data Import*.
- V dialogovém okně ***Data Import*** se vybere y ze seznamu proměnných.
- Dále se zvolí ***Import***.
- V kartě ***Output Data***, se vybere buňka ***Time/Ts***.
- Vybere se ***time*** v dialogovém okně ***Data Import***.
- Kliknutím na tlačítko ***Import*** se provede import časového vektoru pro vstupní data.
- Kliknutím na tlačítko ***Close*** se zavře dialogové okno *Data Import*.

Vykreslení a analýza dat

Po importu estimačních dat je vhodné odstranit odlehlé hodnoty, vyhladit, odstranit trend, nebo jinak ošetřit data tak, aby byla více poddajná pro účely analýzy a hledání odhadu. Pro zobrazení a analýzu vlastností dat je nutné vykreslit časový průběh dat. Pro vykreslení datové sady se vybere v uzlu ***Transient Data*** buňka ***Data*** a klikne se na tlačítko ***Plot Data***.

Pomocí časového průběhu dat je možné zkoumat vlastnosti dat, jako je šum, odlehlé hodnoty a části dat pro použití při hledání odhadu parametrů. Po analýze dat je možné provést předzpracování dat.

2.3.4 Předzpracování dat

Po importu estimačních dat lze provést pomocí nástroje *Data Preprocessing Tool* následující operace pro předzpracování (preprocessing) dat:

- *Exclusion* – Vyloučení části dat z procesu hledání odhadu.
- *Handle missing data* – Odstranění chybějících dat nebo výpočet chybějících dat pomocí interpolace.
- *Handle outliers* – Odstranění odlehlých hodnot.
- *Detrend* – Odstranění průměrných hodnot nebo lineárního trendu.
- *Filter* – Vyhlazení dat pomocí filtru prvního řádu, libovolnou přenosovou funkcí, nebo ideálním filtrem.

Pro otevření nástroj *Data Preprocessing Tool*:

- V GUI se vybere uzel ***Transient Data*** v uzlu ***Estimation Task*** a poté se vyberou data určená pro předzpracování, buď na kartě ***Input Data*** nebo ***Output Data***.
- Klikne se na tlačítko ***Pre-process*** pro otevření nástroje *Data Preprocessing Tool*. Okno nástroje *Data Preprocessing Tool* je na obrázku: 2.3.

2.4 Hledání odhadu parametrů a stavů

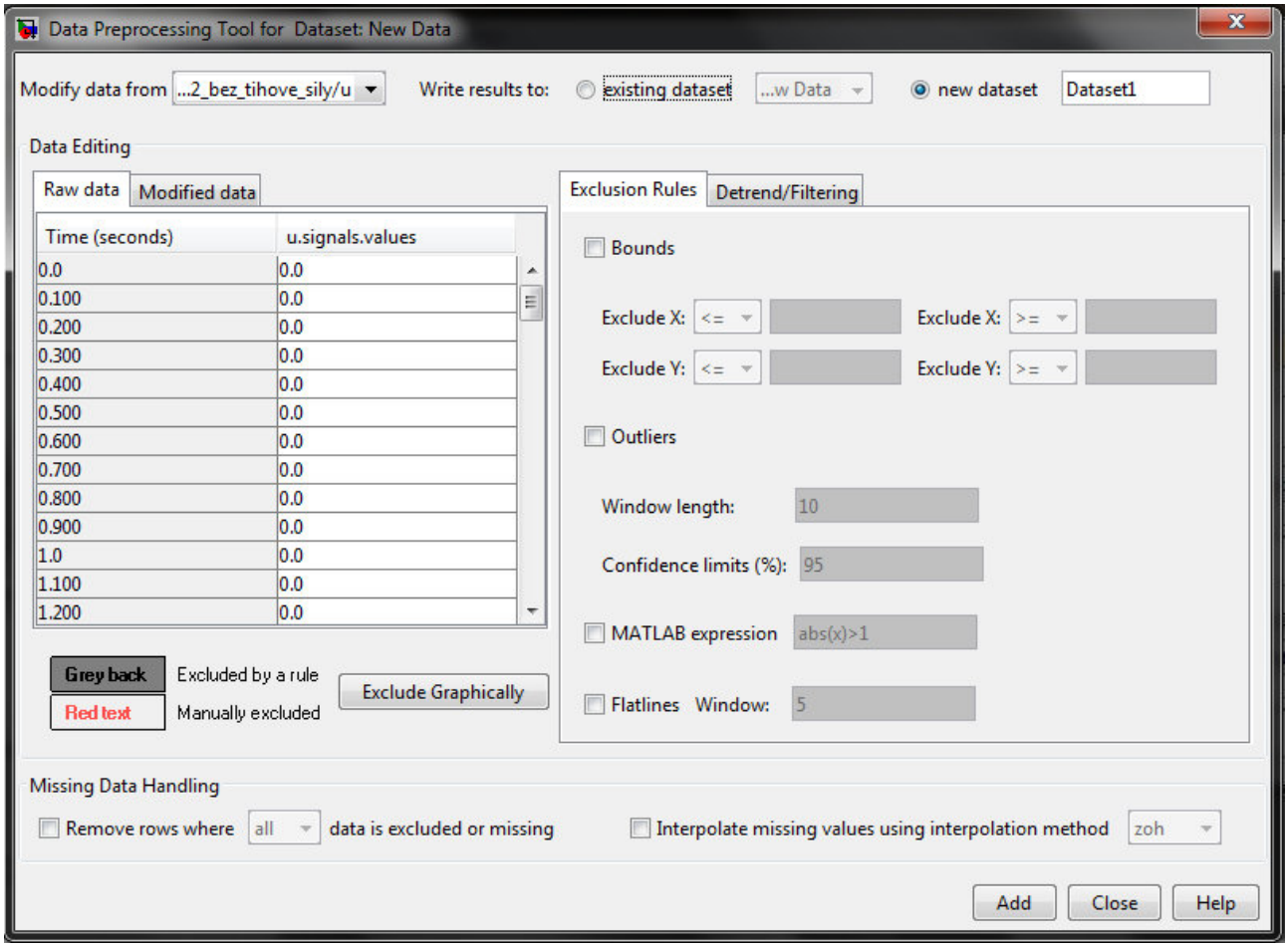
2.4.1 Specifikace estimačních dat

Vytvoření úlohy hledání odhadu

Po importu naměřených signálových dat, jak je popsáno v kapitole 2.3.3 (Import dat), je potřeba vytvořit úlohu hledání odhadu (estimační úlohu) a konfigurovat nastavení hledání odhadu. Pokud data obsahují šum nebo odlehlé hodnoty, je zapotřebí provést předzpracování (preprocessing) dat viz kapitola 2.3.4.

Vytvoření úlohy hledání odhadu:

- V GUI *Control and Estimation Tools Manager* se klikne pravým tlačítkem myši na uzel ***Estimation*** a zvolí se ***New***.
- Vybere se uzel ***New Estimation***.



Obrázek 2.3: Data Preprocessing Tool

Specifikovat data

Po výběru uzlu ***New Estimation*** se objeví záložka ***Data Sets***. Zde lze vybrat datovou sadu určenou pro hledání odhadu.

Poznámka: Pokud je importováno několik datových sad, mohou se vybrat pro hledání odhadu zaškrtnutím políčka vpravo od každé požadované datové sady. Použití více datových sad vede ke zvýšení přesnosti hledání odhadu. Nicméně zvýší se tím také počet potřebných simulací.

Poté je možné určit váhu každého výstupu z tohoto modelu nastavením sloupce ***Weight*** v tabulce ***Output data weights***.

Relativní váhy určují, jak velký důraz je kladen na konkrétní výstupní proměnné. Při definování vah je vhodné řídit se následujícími tipy:

- Použit menší váhu, pokud výstup obsahuje šum
- Použit větší váhu, pokud výstup silně ovlivňuje parametry
- Použit větší váhu, pokud je důležitější, aby tento výstup modelu přesně odpovídal datům

2.4.2 Specifikace odhadovaných parametrů

Volba parametrů, které mají být odhadnuty nejdříve

Hledání odhadu parametrů modelu je iterační proces. Často je praktičtější hledat odhad malé skupiny parametrů a použít finální odhadované hodnoty jako výchozí bod pro další hledání odhadu parametrů, která jsou složitější. Pokud je hledán odhad velkého množství parametrů, je vhodné nejprve hledat odhad parametrů, které nejvíce ovlivňují výstup. Toto rozhodování vyžaduje zkušenosti, intuici a dobrou znalost silných stránek a omezení Simulink modelu. Po vyhledání odhadu podmnožiny parametrů a jejich validaci (verifikaci) se vyhledá odhad zbývajících parametrů.

Specifikovat odhadované parametry

Volba odhadovaných parametrů:

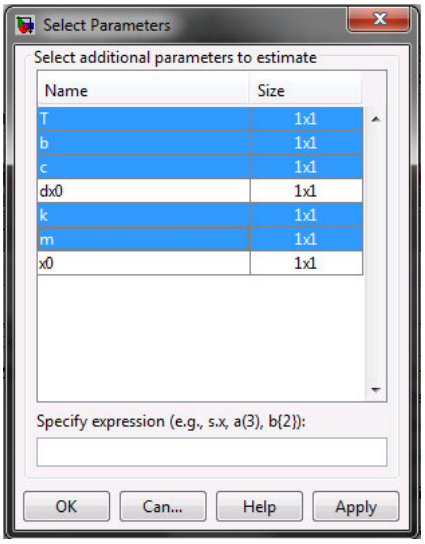
- V GUI *Control and Estimation Tools Manager* se vybere uzel ***Variables*** pro otevření panelu ***Estimated Parameters***.
- V panelu ***Estimated Parameters*** se kliknutím na ***Add*** otevřete dialogové okno *Select Parameters*.

V dialogovém okně (Viz. obrázek: 2.4) je zobrazen seznam všech proměnných modelu. K výběru parametrů pro hledání odhadu lze použít myš nebo textové pole. Pro výběr sousedních parametrů pomocí myši se stiskne a drží klávesa *Shift* a klikne se na první a poslední požadovaný parametr. Pro výběr parametrů, které nejsou umístěny pod sebou, se stiskne a drží klávesa *Ctrl* a je třeba kliknout na každý parametr. Parametry, oddělené čárkami, lze také zadat do textového pole *Specify expression*. Parametry mohou být uloženy v jedné z následujících možností:

- Parametr objektu v Simulinku. Příklad: Pro parametr objektu k se zadá *k.value*.
- Struktura. Příklad: Pro strukturu S, se zadá *S.fieldname* (kde *fieldname* představuje název pole, které obsahuje parametr).
- Cell array. Příklad: Zadáním *C1* pro výběr prvního prvku Cell array C.
- MATLAB array. Příklad: Zadáním *a(1:2)* pro výběr prvního sloupce array a o rozměru 2x2.

Poznámka: Není nutné hledat odhad všech zde vybraných parametrů najednou. Je možné nejprve vybrat všechny parametry, které jsou zajímavé, a později vybrat pouze ty parametry, jejichž odhad má být hledán, jak je popsáno v dalším kroku:

- V uzlu ***New Estimation***, se vybere karta ***Parametry***. V tomto okně můžete vybrat, které parametry mají být odhadovány a rozsah hodnot pro hledání tohoto odhadu.
- Vyberou se parametry, které mají být odhadnuty, zaškrtnutím políčka ve sloupci ***Estimate***.



Obrázek 2.4: Výběr parametrů

– Ve sloupci *Initial Guess* se zadají počáteční hodnoty pro parametry. Výchozí hodnoty ve sloupcích *Minimum* a *Maximum* jsou –Inf a +Inf (tedy -∞ a +∞), ale může se zadat libovolný rozsah.

  Poznámka: Když se zde zadají minimální a maximální hodnoty parametrů, nemá to vliv na nastavení v uzlu Variables. Tyto volby se týkají pouze této úlohy hledání odhadu. Mezi uzly Variables a Estimation lze data přesunout.

Pokud je dobrý důvod se domnívat, že parametr leží v konečném rozsahu, je obvykle nejlepší nepoužívat výchozí minimální a maximální hodnoty. Zadání konečných mezí je obvykle i výpočetně výhodnější. Určení spodní a horní meze může být velmi důležité. Například pokud parametr udává hmotnost, je žádoucí zadat jako absolutní dolní mez hodnotu 0, pokud není k dispozici lepší znalost.

Specifikace počátečních odhadů a horních/spodních mezí

Po výběru odhadovaných parametrů v uzlu ***Variables*** vypadá záložka Estimated Parameters jako na obrázku 2.5.

Pro každý parametr lze použít panel ***Default settings*** pro zadání následujících položek:

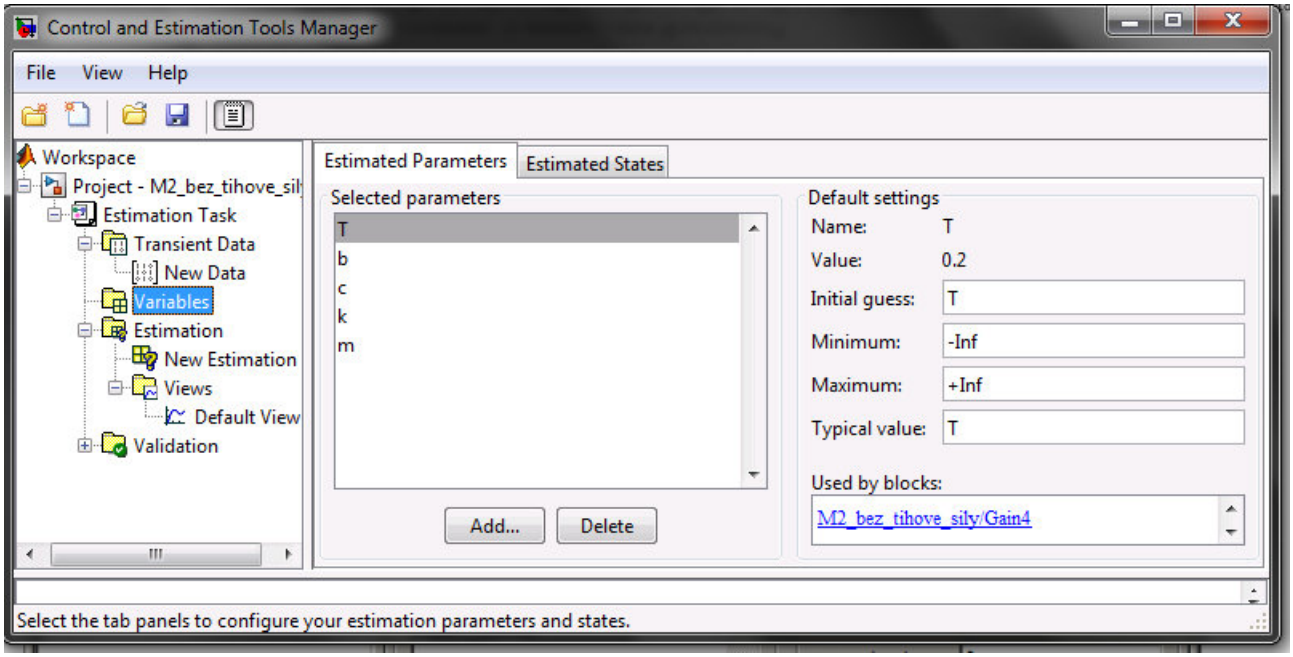
– *Initial guess* (počáteční odhad) – Hodnota, která se použije při zahájení procesu hledání odhadu.

– *Minimum* – Nejmenší povolená hodnota parametru. Výchozí hodnota je –Inf, tedy -∞.

– *Maximum* – Největší povolená hodnota parametru. Výchozí hodnota je +Inf, tedy +∞.

– *Typical value* (typická hodnota) – Průměrný řád. Pokud se předpokládá, že se bude parametr měnit v rozsahu několika řádů, zadá se číslo, které odpovídá průměrnému předpokládanému řádu. Například pokud je počáteční odhad 10, ale očekává se, že se parametr bude měnit mezi 10 a 1000, zvolí se 100 jako typickou hodnotu (průměrný řád).

  Pro kladení většího či menšího důrazu na konkrétní parametry lze použít větší typickou hodnotu, pokud je žádoucí, aby byl na daný parametr kladen větší důraz během hledání jeho odhadu.



Obrázek 2.5: Nastavení parametrů

2.4.3 Specifikace známých počátečních stavů

Specifikace počátečních stavů versus hledání odhadu počátečních stavů

Sady naměřených dat se často shromažďují v různých časech a za různých počátečních podmínek. Když se hledá odhad parametrů modelu pomocí jedné sady dat a následně se spustí další hledání odhadu s druhou sadou dat, hodnoty parametrů se nemusí shodovat. To je samozřejmě problém, vzhledem k tomu, že software PE se snaží najít konstantní hodnoty parametrů.

Hledání odhadu počátečních podmínek je možné s použitím podobných postupů jako při hledání odhadu parametrů. Tyto odhady počátečních podmínek lze poté použít jako základ pro hledání odhadu parametrů Simulink modelu. *Control and Estimation Tools Manager* obsahuje panel ***Estimated States***, kde jsou uvedeny dostupné stavy pro hledání odhadu počátečních podmínek.

Odhadu počátečních podmínek pro bloky s externími počátečními podmínkami

Když blok *Integrator* používá port pro počáteční podmínku, která je specifikována pomocí bloku *IC*, nelze hledat odhad počátečních podmínek integrátoru pomocí softwaru PE. Hledání odhadu není možné, protože externí počáteční podmínky mají vyšší prioritu než počáteční podmínky konkrétního bloku, kvůli zachování integrity modelu.

Pro ladění počátečních podmínek bloku *Integrator* s externími počátečními podmínkami je nutné upravit model tak, aby byl externí signál laditelným parametrem. Například je možné nastavit blok *IC* (který je přiveden do integrátoru) tak, aby to byla laditelná proměnná, a pak lze hledat odhad této proměnné.

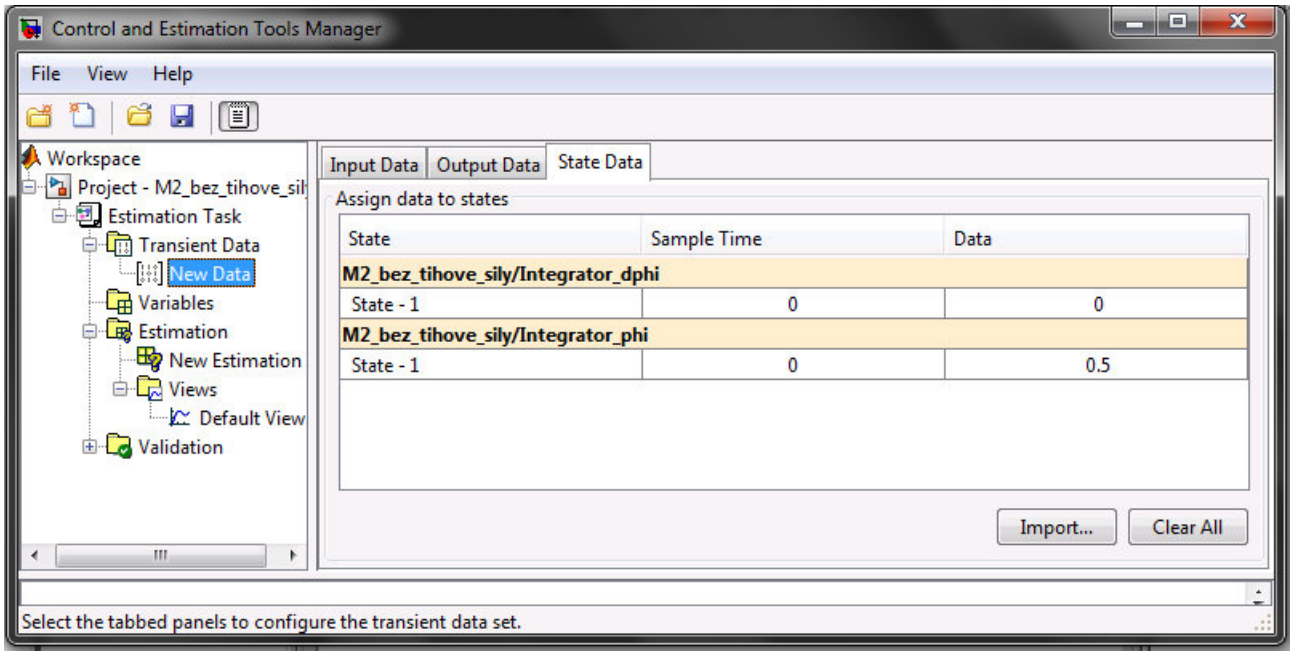
specifikace počáteční stavu v GUI

Po výběru parametrů pro hledání odhadu, jak je popsáno v kapitole Specifikace odhadovaných parametrů (2.4.2), je možné zadat počáteční podmínky stavů v modelu. Ve výchozím nastavení používá hledání odhadu počáteční podmínky stanovené v Simulink modelu. Pokud je žádoucí zadat jiné než výchozí počáteční podmínky, použije se karta ***State Data*** v uzlu ***New Data*** pod uzlem ***Transient Data***.

Pro určení počáteční podmínky stavu pro daný model:

– Vybere se příslušná buňka ***Data***.

– Zadá se počáteční podmínka. Viz. obrázek 2.6.



Obrázek 2.6: Specifikace počátečních stavů

2.4.4 Možnosti hledání odhadu

Pro přístup k možnostem hledání odhadu se klikne v uzlu ***New Estimation*** na kartu ***Estimation***. Následně se klikne na tlačítko ***Estimation Options***. Tato akce otevře dialogové okno *Options*, kde se může na kartě ***Optimization Options*** specifikovat metoda hledání odhadu, možnosti algoritmu a kriteriální funkce pro hledání odhadu. Viz obrázek 2.7.

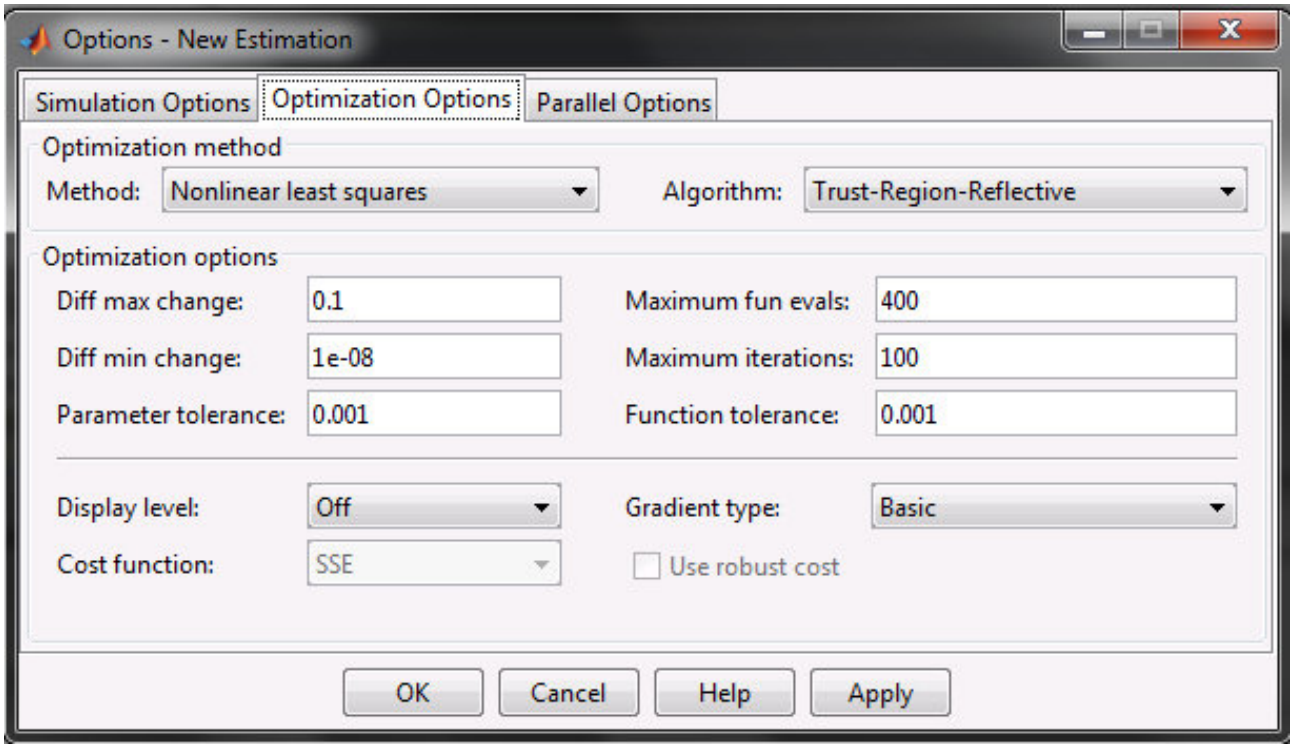
Následující oddíly popisují nastavení metod hledání odhadu a kriteriální funkce:

– Podporované metody hledání odhadu

– Volba možností ukončení optimalizace

– Volba dalších možností optimalizace

– Specifikace kritéria shody (kriteriální funkce)



Obrázek 2.7: Možnosti hledání odhadu

Podporované metody hledání odhadu

Nastavení metody a algoritmu definují optimalizační metodu. Jedná se o oblast ***Optimization method*** v dialogovém okně *Options* pro nastavení metody a algoritmu hledání odhadu.

Pro volbu *Method* jsou k dispozici následující čtyři možnosti:

- *Nonlinear least squares* (výchozí) – Používá *Optimization Toolbox* funkci nelineárních nejmenších čtverců *lsqnonlin*.

  Algoritmy:

  - *Trust-Region-Reflective* (default)
  - *Levenberg-Marquardt*

- *Gradient descent* – Používá *Optimization Toolbox* funkci *fmincon*.

  Algoritmy:

  - *Active-Set* (default)
  - *Interior-Point*
  - *Trust-Region-Reflective*
  - *Sequential Quadratic Programming*

- *Pattern search* – Používá metodu hledání vzorů *patternsearch*. Tato možnost vyžaduje software *Global Optimization Toolbox*.

- *Simplex search* - používá *Optimization Toolbox* funkci *fminsearch*, což je přímá metoda vyhledávání. *Simplex search* je nejužitečnější pro jednoduché problémy a je někdy rychlejší než *fmincon* u modelů, které obsahují nespojitosti.

Volba možností ukončení optimalizace

Možnosti ukončení optimalizace se stanovují v oblasti ***Optimization options***.

Ukončení optimalizace je definováno několika volbami:

- *Diff max change* – Maximální povolená změna proměnných pro derivace konečných diferencí.
- *Diff min change* – Minimální povolená změna proměnných pro derivace konečných diferencí.
- *Parameter tolerance* – Optimalizace se ukončí, když se po sobě jdoucí hodnoty parametrů změní o méně než toto číslo.
- *Maximum fun evals* – Maximální povolený počet ohodnocení kriteriální funkce. Optimalizace se ukončí, když počet ohodnocení funkce překročí tuto hodnotu.
- *Maximum iterations* – Maximální povolený počet iterací. Optimalizace se ukončí, pokud počet iterací překročí tuto hodnotu.
- *Function tolerance* – Optimalizace se ukončí, když po sobě jdoucí hodnoty funkce jsou menší než tato hodnota.

Změnou těchto parametrů lze přinutit optimalizaci k pokračování v hledání řešení nebo k pokračování v hledání přesnějšího řešení.

Volba dalších možností optimalizace

V dolní části panelu ***Optimization options*** je skupina dalších možností optimalizace.

Další možnosti pro optimalizaci zahrnují:

- *Display level* – Určuje formu výstupu, který se zobrazí v *MATLAB command window*.

  Možnosti jsou:

  - *Iteration* – zobrazení informací po každé iteraci.
  - *None* – žádný výstup.
  - *Notify* – zobrazení výstupu pouze v případě, že funkce nekonverguje.
  - *Final* – zobrazení pouze konečného výstupu.

- *Gradient type* – Při použití metod *Gradient descent* nebo *Nonlinear least squares* se gradienty vypočítávají na základě metody konečných diferencí. Metoda *Refined* poskytuje robustnější a méně šumem zatíženou metodu výpočtu gradientu než *Basic*, nicméně optimalizace pomocí metody *Refined* trvá déle.

Specifikace kritéria shody (kriteriální funkce)

Kriteriální funkce je funkce, kterou se metody hledání odhadu snaží minimalizovat. Kriteriální funkci lze specifikovat v dolní části oblasti ***Optimization options***.

Při výběru kriteriální funkce jsou následující možnosti:

– *Cost function* – Výchozí je *SSE* (součet čtverců odchylek), který používá přístup nejmenších čtverců. Je možné také použít *SAE* (součet absolutních odchylek). Více v kapitole 2.2.2.

– *Use robust cost* – Optimalizátor používá robustní kriteriální funkci namísto výchozího kritéria nejmenších čtverců. To je užitečné, pokud mají experimentální data mnoho odlehlých hodnot, nebo pokud obsahují šum.

2.4.5 Možnosti simulace

K hledání odhadu parametrů modelu software PE spouští simulaci modelu.

Pro přístup k možnostem simulace se klikne v uzlu ***New Estimation*** na kartu ***Estimation***. Kliknutím na tlačítko ***Estimation Options***. Tato akce otevře dialogové okno *Options*, kde je možné v kartě ***Simulation Options*** specifikovat možnosti simulace (čas simulace, řešiče), popsané v následujících oddílech.

Volba času simulace

Čas začátku a konce simulace se může nastavit v oblasti ***Simulation time*** na kartě ***Simulation Options***.

Ve výchozím nastavení jsou časy začátku simulace (*Start time*) i konce simulace (*Stop time*) automaticky vypočteny na základě počátečních a koncových časů simulace nastavených v Simulink modelu.

Pro nastavení jiného počátečního a koncového času pro optimalizaci, se zadají nové časy v oblasti ***Simulation time***. Tato akce přepíše počáteční a koncové časy simulace specifikované v Simulink modelu.

Volba řešičů

Při hledání odhadu software řeší dynamický systém pomocí jednoho z několika řešičů v Simulinku.

Určí se typ řešiče a jeho možnosti v oblasti ***Solver options*** na kartě ***Simulation Options*** v dialogovém okně *Options*.

Řešič může být jedním z následujících typů (***Type***):

– *Auto* (výchozí) – Používá nastavení simulace specifikované v Simulink modelu.

– *Variable-step* – Řešiče s proměnným krokem udržují chybu uvnitř stanovených tolerancí úpravou velikosti kroku, kterou řešič používá. Například pokud se stavy modelu budou pravděpodobně rychle měnit, lze použít řešič s proměnným krokem pro rychlejší simulaci.

  Řešiče (***Solver***):

  – *Discrete* (no continuous states)
  – *ode45* (Dormand-Prince)
  – *ode23* (Bogacki-Shampine)
  – *ode113* (Adams)
  – *ode15s* (stiff/NDF)

  - *ode23s* (stiff/Mod. Rosenbrock)
  - *ode23t* (Mod. stiff/Trapezoidal)
  - *ode23tb* (stiff/TR-BDF2)

- *Fixed-step* – Řešiče s fixním krokem používají konstantní velikost kroku.
  Řešiče (***Solver***):
  - *Discrete* (no continuous states)
  - *ode5* (Dormand-Prince)
  - *ode4* (Runge-Kutta)
  - *ode3* (Bogacki-Shanpine)
  - *ode2* (Heun)
  - *ode1* (Euler)

Poznámka: Pro rychlejší simulace při hledání odhadu je možné změnit typ řešiče na *Variable-step* nebo *Fixed-step*. Nicméně odhadnuté hodnoty parametrů platí pouze pro vybraný typ řešiče a mohou se lišit od hodnot získaných pomocí nastavení specifikovaných v Simulink modelu.

je možné také zadat následující parametry, které mají vliv na velikost kroku simulace:

- *Maximum step size* – Největší velikost kroku, kterou může řešič použít během simulace.
- *Minimum step size* – Nejmenší velikost kroku, kterou může řešič použít během simulace.
- *Initial step size* – Velikost kroku, kterou řešič používá k zahájení simulace.
- *Relative tolerance* – Největší přípustná relativní chyba v každém kroku simulace.
- *Absolute tolerance* – Největší přípustná absolutní chyba v každém kroku simulace.
- *Zero crossing control* – Při nastavení na *on* řešič přesně spočítá místo, kde signál protíná osu x. Tato volba je užitečná při používání funkcí, které jsou nehladké, a výstup závisí na tom, kdy signál protíná osu x, například absolutní hodnoty.

  Ve výchozím nastavení software automaticky zvolí hodnoty těchto voleb.

2.4.6 Spuštění hledání odhadu

Než se začne s hledáním odhadu parametrů, musí být nakonfigurovány estimační data a parametry a specifikovány možnosti hledání odhadu a simulace, což bylo popsáno v předcházejícím textu.

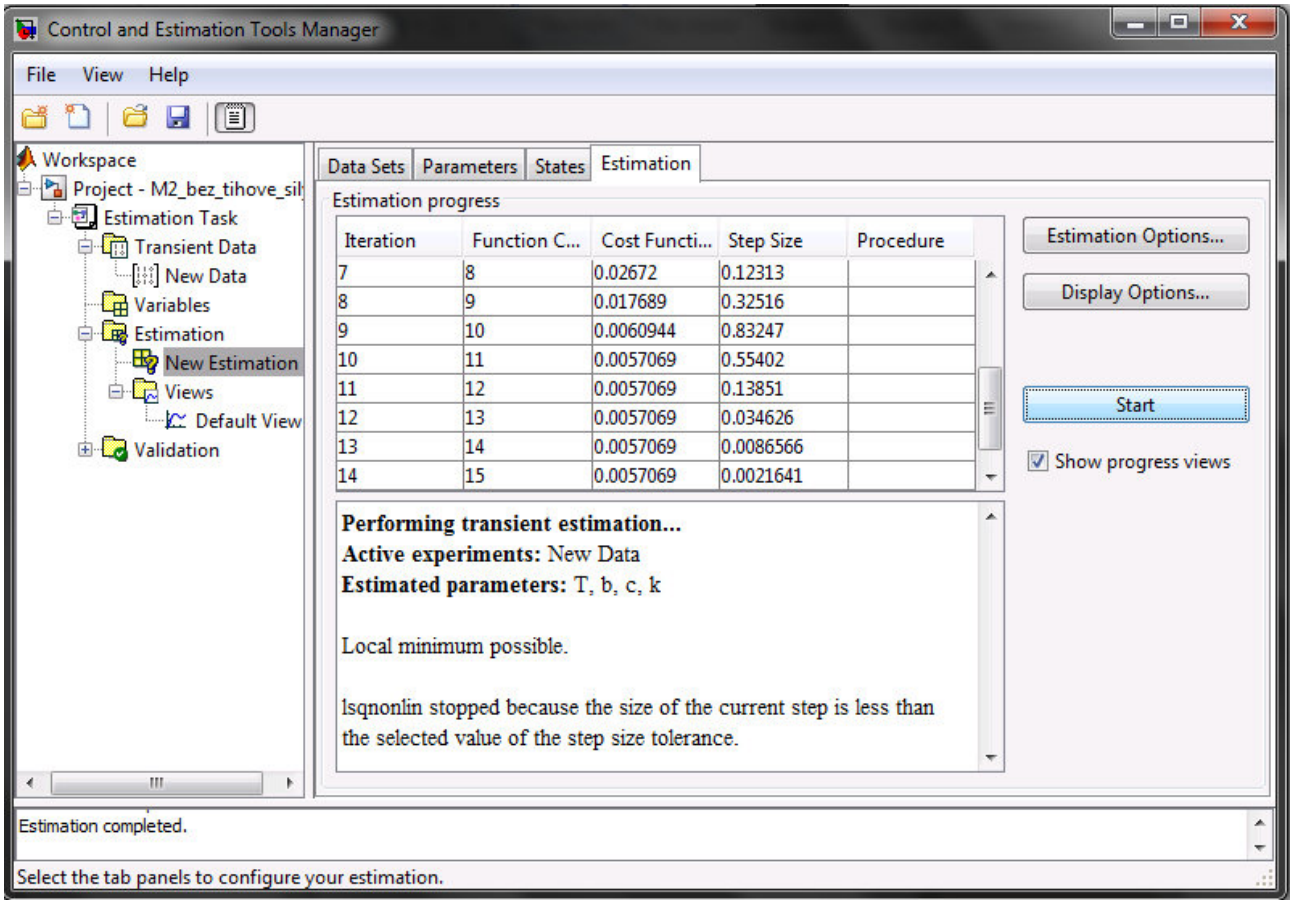
Pro spuštění hledání odhadu se vybere uzel ***New Estimation*** a vybere se karta ***Estimation***. Stisknutím tlačítka ***Start*** se zahájí proces hledání odhadu. Po ukončení iterací by okno mělo vypadat podobně jako na obrázku 2.8.

Nižší hodnota kriteriální funkce (*Cost function*) obvykle značí úspěšné nalezení odhadu, což znamená, že experimentální data odpovídají simulaci modelu s odhadnutými parametry.

Karta ***Estimation*** zobrazuje každou iteraci optimalizačních metod. Pro zobrazení finálních hodnot parametrů je potřeba se přepnout na kartu ***Parameters***.

Hodnoty těchto parametrů jsou také aktualizovány v *MATLAB workspace*. Pokud zadáte název proměnné ve sloupci ***Initial Guess***, můžete restartovat hledání odhadu z místa, kde skončilo předchozí hledání odhadu.

Dále je možné prohlédnout graf porovnání naměřených a simulovaných dat s cílem zjistit, jak dobře simulovaná data odpovídají naměřeným estimačním datům viz. obrázek 2.9.



Obrázek 2.8: Ukončení hledání odhadu

2.4.7 Typy grafů

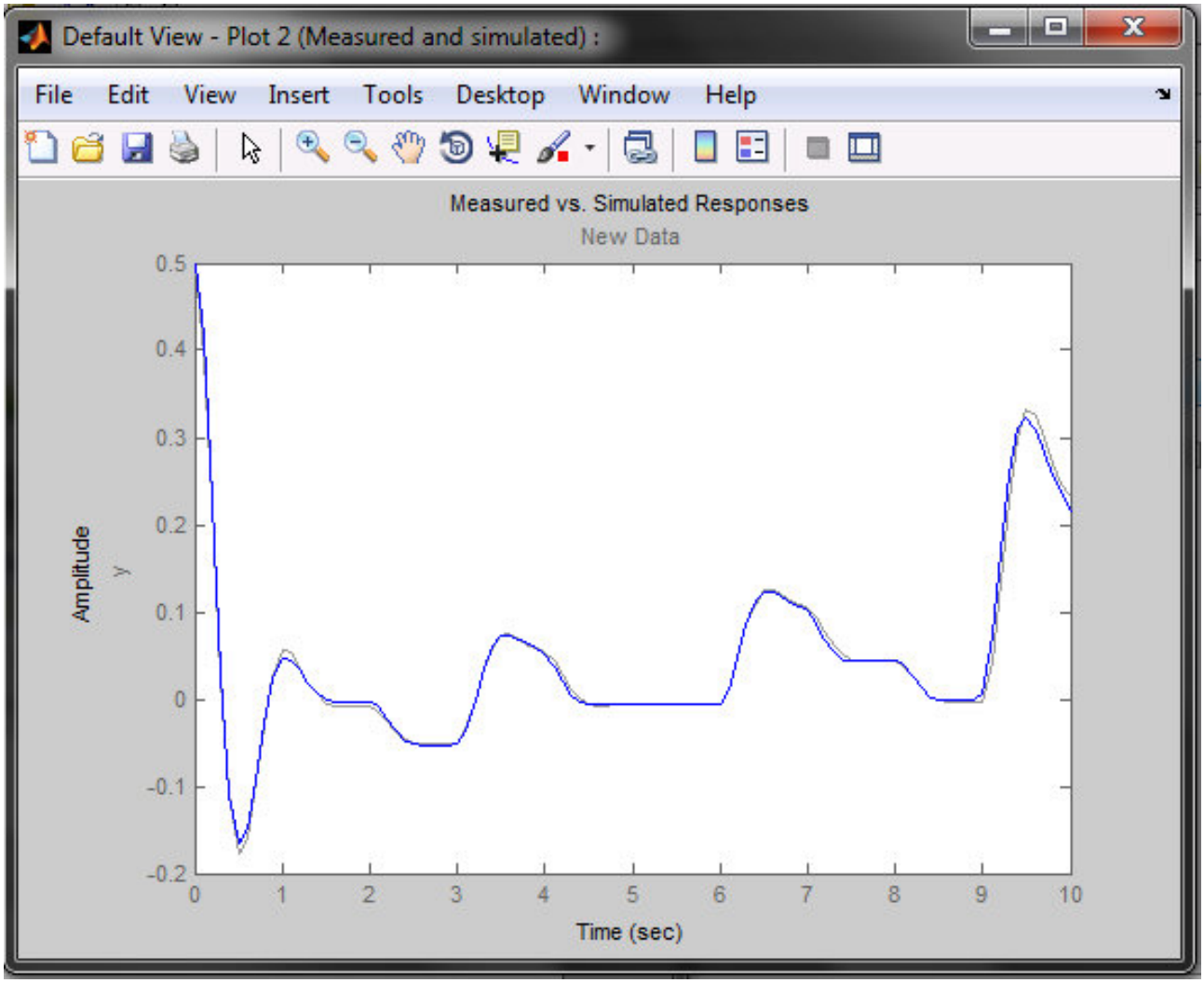
Uzel ***Views*** slouží k prohlížení a vyhodnocení hledání odhadu. Typ grafu lze zvolit v rozbalovacím seznamu v kolonce ***Plot Type***. K dispozici jsou následující typy grafů:

- *Cost function* – Graf hodnot kriteriální funkce.
- *Measured and simulated* – Graf porovnání naměřených a simulovaných dat.
- *Parameter sensitivity* – Graf rychlosti změny kriteriální funkce jako funkce změny parametru. Je to tedy graf derivace kriteriální funkce s ohledem na měnící se parametr.
- *Parameter trajectory* – Graf měnících se hodnot parametru.
- *Residuals* – Graf odchylky mezi experimentálními daty a simulovaného výstupu.

2.4.8 Validace modelu

Po dokončení hledání odhadu parametrů, jak je popsáno v kapitole Spuštění hledání odhadu (2.4.6), je nutné ověřit (validovat, verifikovat) výsledky pomocí jiné sady dat (validační datová sada).

Postup pro validaci modelu:



Obrázek 2.9: Graf porovnání naměřených a simulovaných dat

- Import validační datové sady v uzlu ***Transient Data***.
- Přidání nové validační úlohy v uzlu ***Validation***.
- Konfigurace nastavení validace - výběr typů grafu a validační datové sady v záložce ***Validation Setup***.
- Zvolení položky ***Show Plots*** v záložce ***Validation Setup*** a zobrazení výsledků v grafu.
- Porovnání validačních grafů s odpovídajícími estimačními grafy (uzel ***Views***).

Základní rozdíl mezi rysy uzlů ***Validation*** a ***Views*** je ten, že validaci lze spustit až po dokončení hledání odhadu. Všechny náhledy (*views*) by měly být vytvořeny před hledáním odhadu a aktualizace náhledů je možné pozorovat v reálném čase. Validace mohou použít i jiné validační datové sady pro porovnání s odezvou modelu. Navíc validace se objeví až po dokončení hledání odhadu a neaktualizují se.

Validovat data je možné porovnáním naměřených a simulovaných dat pro estimační data a validační datové sady. Stejným způsobem je také často užitečné porovnávat reziduály.

2.4.9 Urychlení simulací modelu během hledání odhadu

Úvod o urychlení simulací modelu během hledání odhadu

Výpočty hledání odhadu parametrů lze urychlit změnou simulačního režimu (*simulation mode*) Simulink modelu. Software PE podporuje simulační režimy *Normal* a *Accelerator*.

Výchozí režim simulace je *Normal*. V tomto režimu Simulink používá během simulací interpretovaný kód, nikoliv tedy zkompilovaný kód v jazyce C.

V režimu *Accelerator* software PE při hledání odhadu spouští simulace se zkompilovaným kódem v jazyce C. Použití zkompilovaného kódu v jazyce C urychluje simulace a zkracuje čas pro hledání odhadu parametrů.

Omezení

Režim *Accelerator* není možné použít, pokud model obsahuje algebraické smyčky. V případě, že model obsahuje bloky *MATLAB function*, je nutné je buď odstranit nebo je nahradit bloky *Fcn*.

Nastavení režimu Accelerator pro hledání odhadu parametrů

Pro nastavení simulačního režimu *Accelerator* se v Simulink modelu provede jedna z následujících akcí:

- Vybere se *Simulation - Mode - Accelerator*.
- Volba *Accelerator* v rozbalovacím seznamu vedle času konce simulace.

Pro dosažení maximálního výkonu v režimu *Accelerator* je vhodné zavřít všechny bloky *Scope* v modelu.

Urychlení pomocí paralelních výpočtů

Software PE lze použít se softwarem *Parallel Computing Toolbox* pro urychlení hledání odhadu parametrů Simulink modelů. Použití paralelních výpočtů může zkrátit dobu hledání odhadu v následujících případech:

- Model obsahuje velké množství odhadovaných parametrů a metoda hledání odhadu je nastavena jako *Nonlinear least squares* nebo *Gradient descent*.
- Metodou hledání odhadu je zvolena metoda *Pattern search*.
- Model je složitý a jeho simulace trvá dlouhou dobu.

Při použití paralelních výpočtů software distribuuje nezávislé simulace, aby je mohl spouštět paralelně v několika relacích MATLABu, známé jako *workers*. Čas potřebný pro simulaci modelu dominuje celkovému času hledání odhadu. Z toho důvodu distribuce simulací významně snižuje čas hledání odhadu.

2.5 Zhodnocení nástroje Parameter Estimation

Program PE je silný nástroj, který umožňuje hledání parametrů a stavů modelu. Program je velice komplexní a dovoluje nastavit různé parametry, které mohou přispět k lepšímu hledání odhadu parametrů. Některé funkce jsou ale dostupné jen pokud se s nástrojem pracuje pomocí příkazů. Program má také několik nedostatků.

Seznam nedostatků:

- Před hledáním odhadu je nutno zadat nějaké hodnoty parametrů do *Workspace*, i když jejich hodnotu budeme estimovat (hledat její odhad). Pokud hodnoty parametrů ve *Workspace* nejsou specifikovány, nástroj PE zahlásí chybu.
- Data uložená ve formátu struktury nelze importovat pomocí dialogového okna, ale je potřeba je napsat do buňky Data manuálně (např. *u.signals.values*; *u.time*).
- Problém s konvergencí optimalizačních metod – časté uvíznutí v lokálním minimu, je to velmi citlivé na počáteční podmínky (počáteční odhad).
- Program pracuje vždy jen s jedním modelem.
- Program neukládá výsledky simulací pro pozdější vizualizaci.
- Nelze nastavit minimální počet iterací nebo dobu hledaní parametrů.
- Program je placený a není součástí studenské verze programu Matlab, ani do studenské verze nelze doinstalovat.
- Pro podrobnější definici hledání odhadu parametrů je potřeba pracovat s programem pomocí příkazů.

3 Cíle řešení

Pro program MPE (Mechlab's parametr estimation) se na základě zhodnocení programu PE definovaly požadavky, které by měl program splňovat. Požadavky lze rozdělit na systémové požadavky, vizuální dovednosti a na schopnosti programu. Na základě definice bude vytvořen program, který by měl mít stejné základní vlastnosti, jako program PE, a přitom přinášet nové možnosti pro uživatele, které mu program od Mathworks neumožňuje.

3.0.1 Schopnosti, definované pro program

- **ukládaní průběhu simulací** - Po každé simulaci se kromě vypočítané chyby uloží i průběh odezvy ze simulace, aby posloužila pro vizualizaci nebo pro výpočet hodnoty chyby jinou metodou.
- **možnost kombinovat více prohledávacích metod** - Uživatel by měl mít možnost pokračovat v započatém prohledávání jinou metodou nebo stejnou metodou s jiným nastavením.
- **možnost jednoduchého přidání nové prohledávací metody** - Nová metoda by se měla přidat pouhým vložením do složky s ostatními metodami.
- **volba metodiky určování chyby** - Pro výpočet ohodnocení parametrů by mělo být možné použít více metodik výpočtu, například: MAE, RMSE,...
- **zohlednit důvěru pro jednotlivé experimenty** - Zadaná hodnota pro danou kombinaci modelu a souboru dat (experimentu). Chyby jsou touto váhou násobeny, a tím se význam některých experimentů potlačí nebo zvýrazní.
- **správa projektů, souborů dat a modelů** - Přehledná souborová politika.
- **nastavitelná tolerance shody navržených parametrů** - Zabránit simulacím, které se v prohledávacím prostoru pohybují blízko k dříve prohledanému místu.
- **iterační zmenšování prohledávaného prostoru změnou hranic intervalu** - Uživatelsky přívětivá práce s hranicemi intervalů.
- **porovnání jednotlivých modelů nebo souborů dat** - Možnost zhodnotit, jak vytvořené modely a naměřená data reprezentují daný experiment.
- **porovnání prohledávacích metod** - Možnost spustit více na sebe nezávislých prohledávacích metod se stejnými počátečními podmínkami umožní porovnání jejich schopností.
- **možnost převzorkovat data** - Možnost změnit vzorkovací frekvenci naměřených dat.
- **možnost pracovat s programem pomocí příkazů** - Vytvořit logickou syntaxi funkcí a umožnit tak pracovat s programem i bez uživatelského rozhraní.

- **schopnost najít parametry v modelu, i když jsou součástí vzorce** - Některé parametry nemusí být v bloku osamostatněné. Mohou být součástí matematického vzorce, například Mz/J. Program parametry musí umět najít.
- **možnost pracovat s více modely, které sdílejí parametry** - Program MPE by měl na rozdíl od programu PE umět pracovat s více modely.
- **přidání nalezených parametrů do modelu** - Schopnost programu upravit model tak, aby byl spustitelný bez přepisování nalezených parametrů.
- **možnost spojit více estimací** - Aby se mohla práce paralelizovat na více počítačů, je potřeba mít možnost prohledávání spojit.
- **oříznutí kombinace nalezených parametrů** - Možnost zmenšit soubor pomocí zadaného kritéria nalezených parametrů.
- **odlišit konstanty a parametry** - Některé prohledávací metody pracují s počtem neznámých parametrů, a proto je potřeba odlišit hledané parametry a známé konstanty.

3.0.2 Vizuální požadavky na program

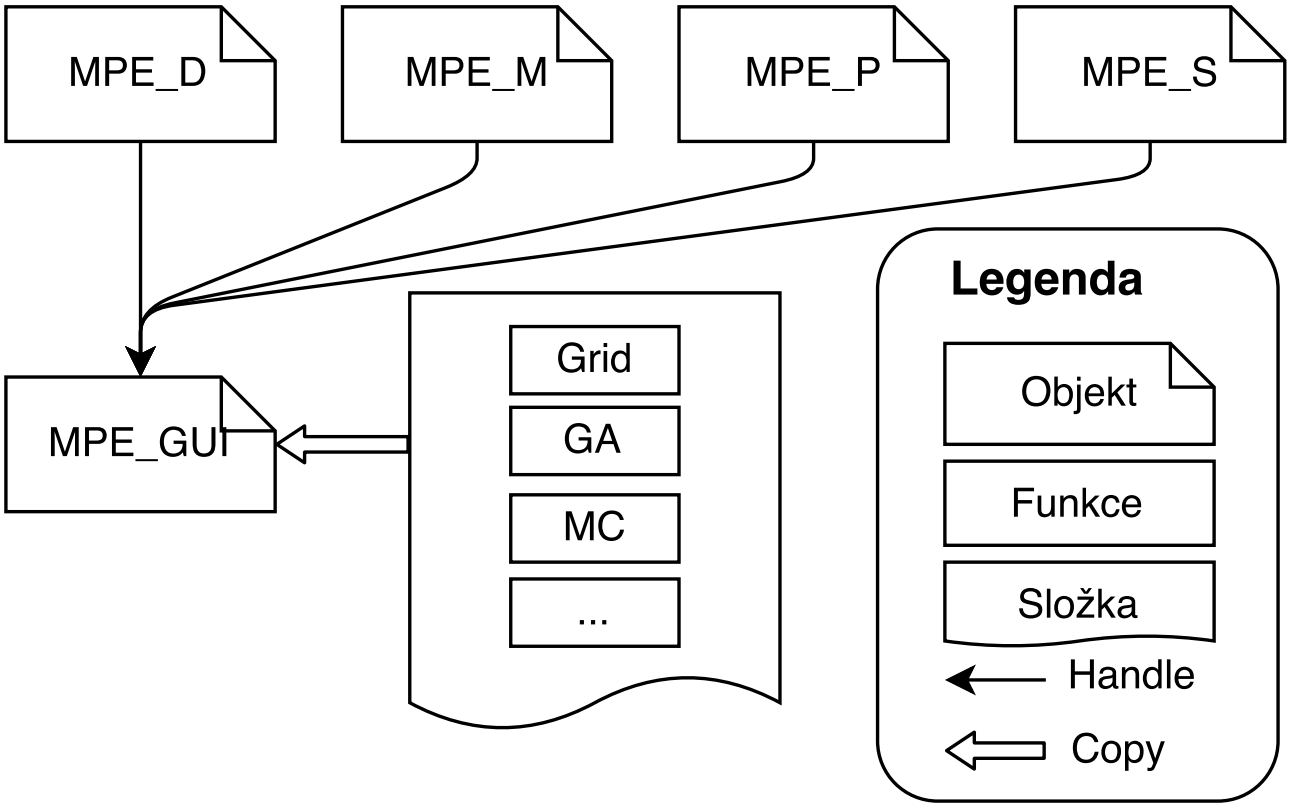
- **intuitivní uživatelské rozhraní**
- **možnost vizualizace nalezených parametrů a odezvy systému během prohledávání** - Pro optimalizaci prohledávající metody je vhodné vidět jak navržené parametry, tak i odezvu ze simulace.
- **porovnání jednotlivých modelů nebo souborů dat** - Vizualizace nalezených chyb pro jednotlivé kombinace.
- **porovnání prohledávacích metod**
- **vizualizace nalezené chyby v průběhu hledání**
- **vizualizace odezvy pro parametry v určitém intervalu** - Vykreslí se jen odezvy pro parametry, které se nachází v zadaném intervalu.
- **vizualizace hledaných parametrů, které náleží do zadaných intervalů** - Zobrazí se jen ty hodnoty parametrů, které náleží do všech intervalů.
- **vizualizace doby do konce prohledávacího cyklu** - Odhad doby, která zbývá do konce prohledávacího cyklu.
- **responzivní okno programu** - Možnost měnit velikost nebo maximalizovat okno programu.
- **vizualizace vstupního signálu** - Při volbě souborů dat poskytnout možnost vizualizovat vstupní signál soustavy a naměřenou odezvu.
- **informační panel** - Spodní proužek, který doprovází uživatele v definici problému

3.0.3 Systémové požadavky na program

- **kompatibilita od verze Matlab 2012b**
- **fungování bez dalších toolboxů kromě Simulinku** - Program PE je součástí balíčku *Simulink Design Optimization*. Program MPE by neměl využívat žádný doplňkový balíček.
- **kompatibilita pro operační systémy Windows a OS X** - Operační systémy se liší v cestách k souborům a ve formátech, se kterými umí pracovat. V některých částech programu je potřeba ošetřit tyto rozdíly.
- **spustitelnost na studenské verzi Matlabu** - Doplňkový balíček *Simulink Design Optimization* není možné nainstalovat na studenskou verzi Matlabu. Program MPE by měl narozdíl od programu PE fungovat i na studenské verzi Matlabu.

4 Postup řešení a výsledky

4.1 Struktura programu



Obrázek 4.1: Struktura programu

Program je strukturovaný tak, aby pokud možno oddělil jednotlivé problematiky, jako je práce s daty, modelem, výsledky a uživatelským rozhraním. Viz obrázek 4.1. Z tohoto důvodu je program tvořen z pěti objektů, které mají na starosti dříve zmiňované problematiky. Účelem této struktury je co největší přehlednost, protože názvy syntaxe metod, které objekty obsahují, jsou složeny z odkazu na daný objekt a názvu metody, která je za tečkou.

Programování pomocí OOP (objektově orientovaného programování) je programovací přístup, který dovoluje spojit data programu, související činnosti a operace do logických struktur. Tyto struktury se nazývají objekty a jejich funkce metodami. Tento přístup velice zlepšuje schopnost zvládat složitost pokročilejších programů, vzhledem ke schopnosti zvládnutí rozšiřovaní a udržování sofistikovaných programů a složitých datových struktur.[2]

OOP schopnosti jazyka Matlabu vám umožní vytvořit komplexní technické počítačové aplikace rychleji, než s jinými jazyky, jako je C ++ a Java. Je možné definovat třídy a aplikovat standardní návrhové vzory objektově orientované v prostředí Matlabu, které umožňují opětovné

použití kódu, dědičnost a zapouzdření.[2]

Program obsahuje kromě objektů začínajících na písmena MPE a metod ve složce */methods* také funkci *strsplit*. Tato funkce je obsažena v Matlabu od verze R2013a. Protože byl požadavek kompatibility od verze Matlab R2012b, je tato funkce přidána externě. Funkce slouží pro rozdělení textového řetězce pomocí jiného textového řetězce.[2] Pokud je používána verze Matlabu R2013a nebo novější, je vhodné funkci pro urychlení programu odstranit.

4.1.1 Administrace dat

O práci s datovou strukturou se stará objekt MPE_D (Mechlab's parametr estimation - data administration). Zahrnuje funkci, která pomocí cesty k souboru s příponou *.mat* nahraje data do *properties* objektu. Ostatní objekty přistupují k datům pomocí odkazu. Tento přístup zrychluje program, protože předchází přesouvání velkého objemu dat. Objekt zahrnuje další operace s datovou strukturou, které jsou popsány v uživatelském manuálu v příloze: A.1.

4.1.2 Administrace modelů

Objekt MPE_M zajišťuje cestu k modelům a názvy parametrů a konstant v modelech obsažené, které si z modelů sám zjistí. Práce s objektem a jeho metodami je popsána v uživatelském manuálu v příloze: A.1.

4.1.3 Definice parametrů

Parametry, které bude MPE (Mechlab's parametr estimation) hledat a také konstanty, které se během hledání odhadu parametrů nemění, avšak v modelu jsou obsažené, obsahuje objekt MPE_P (Mechlab's parametr estimation - definici parametrs). Objekt také obsahuje rozsahy pro hledání odhadu jednotlivých parametrů. Interval se skládá z minimální a maximální hodnoty rozsahu. Pokud je minimální a maximální hodnota rozsahu stejná, považuje program daný parametr za konstantu a je vyloučen z výpočtů jednotlivých prohledávacích metod, které často pro nastavení svých parametrů používají počet hledaných parametrů. Jako například genetický algoritmus, Grid a další. Práce s objektem a jeho metodami je popsána v uživatelském manuálu v příloze: A.1.

4.1.4 Administrace simulací a jejich vizualizace

Určení, které kombinace modelů a dat poslouží pro hledání odhadu parametru, se definuje pomocí objektu MPE_S (Mechlab's parametr estimation - saving searched prametres and their simulations). V tomto objektu se taky uchovává průběh všech simulací, které jsou pro dané hledání odsimulovány, chyba pro danou kombinaci (rozdíl mezi odezvou naměřenou a odsimulovanou) a také celková chyba ze všech použitých kombinací dat a modelů. V tomto objektu je celá řada metod, které zajišťují práci s výsledky a jejich vizualizace. Práce s objektem a jeho metodami je popsána v uživatelském manuálu v příloze A.1.

4.1.5 Definice metody prohledávání a její parametry

Pro spuštění simulace pomocí uživatelského rozhraní nebo pomocí skriptu slouží objekt MPE_GUI (Mechlab's parametr estimation - Graphical User Interface). Do objektu vstupují objekty MPE_D, MPE_M, MPE_P, MPE_S pomocí odkazů. Vstupem je také název prohledávací metody a libovolný počet parametrů dané metody. Kompletní syntaxe je podrobně popsána v uživatelském manuálu v příloze A.1.

Objekt provede různou posloupnost operací podle toho, jestli je spuštěn jako uživatelské rozhraní, nebo pomocí skriptu. Z tohoto důvodu jsou některá uživatelská rozhraní při druhé možnosti nepřístupná. Jedná se o *uipanel* (vrchní rozklikací panel).

Objekt MPE_GUI také obsahuje ohodnocovaní funkci. Tato funkce načte parametry a konstanty, které jí pošle prohledávací metoda. Načte vstupní data z objektu MPE_D a odsimuluje model v Simulinku, výstup ze simulace porovná s měřením pomocí zvolené porovnávací metody v objektu MPE_S.

4.1.6 Metody prohledávání

Všechny metody prohledávání jsou uložené ve složce */methods*. Jednotlivé metody mohou mít libovolný název (podle pravidel Matlabu), ale musí obsahovat funkci s názvem *m_solve* s přesně definovaným počtem vstupů. Nová metoda se nemusí nikde definovat, postačí přidat danou metodu do této složky a program si ji sám přidá mezi metody prohledávaní. Také zjistí, jaké parametry může uživatel nastavit pro danou metodu.

Metody jsou psané jako funkce, to znamená, že na rozdíl od objektů, které začínají písmeny MPE, nemají svoji datovou strukturu. K datům přistupují pomocí odkazů do jednotlivých objektů. Metody také mohou využívat zobrazovací funkce, které jsou obsažené v objektu MPE_GUI a taky pracují s výsledky, obsaženými v objektu MPE_S. Objekt MPE_GUI při spuštění zkopíruje všechny funkce, obsažené v metodách. Od té chvíle se chovají jako součást objektu MPE_GUI. Z toho plyne, že při úpravě prohledávací funkce je potřeba znovu vytvořit objekt MPE_GUI, aby se provedené změny projevily.

Snahou je, aby přidání nové metody nevyžadovalo hlubokou znalost programu MPE a pro vytvoření nové metody byly dostačující informace, nabyté z uživatelského návodu k programu A.1. Tento přístup umožňuje uživateli experimentovat s různými metodami prohledávání a porovnání s dříve vytvořenými metodami bez nutnosti znalosti komunikace Matlabu a Simulinku a dalších netriviálních operací, které program MPE zajišťuje.

4.1.7 Struktura projektů

Při práci s programem MPE pomocí uživatelského rozhraní je uživatel nucen vytvořit projekt. Při vytvoření projektu se založí složka s jeho názvem ve složce *Projects* a podsložky *Projects/NazevProjektu/Data* a *Projects/NazevProjektu/Models*, do kterých se ukládají kopie dat a modelů.

Vytváření projektů umožňuje zachovávat přehlednost prohledávání a také změnu modelu a jeho nastavení se zachováním originálu. Program provede při překopírovaní modelu i jeho přejmenování. To zabraňuje komplikacím při otevření originálního modelu během simulace.

4.2 Struktura dat

Jak už bylo napsáno v kapitole 4.1, data jsou rozdělena do objektů, začínajících písmeny MPE. V této kapitole bude popsána struktura dat v jednotlivých objektech.

4.2.1 Data (MPE_D)

Data jsou ukládána do tabulky *table* v jednom řádku, kde sloupce jsou jednotlivé datové struktury, které obsahují název dat *name*, časovou stopu *time* (1:počet časových záznamů,1), vstupní hodnoty pro simulaci *u* (1:počet časových záznamů,1), naměřená data *y* (1:počet časových záznamů,1) a cestu k datové struktuře *path*.

```
>> hD.table{1,1}
    name: [1x38 char]
    time: [101x1 double]
    u: [101x1 double]
    y: [101x1 double]
    path: [1x45 char]
```

4.2.2 Modely (MPE_M)

Cesty k modelům jsou uloženy v tabulce *table* v jednom řádku, kde sloupce jsou jednotlivé struktury, které obsahují cestu k modelu *name*. Cesta je uvedena od kořenového adresáře systému. V cestě je také obsažen název modelu, který lze získat pomocí funkce `strsplit(hM.table{1,1}.table.name,'/')`. Dále struktura obsahuje názvy jedinečných *parametrů* (Pokud se parametr objevuje v modelu vícekrát, je uložen jen jednou).

```
>> hM.table{1,1}
    name: [1x58 char]
    parameters: {1x7 cell}
```

4.2.3 Parametry (MPE_P)

Objekt MPE_P obsahuje strukturu, ve které je obsažena tabulka *names*. V ní se nachází v jednom řádku názvy parametrů a parametry jsou ve stejném sloupci, jako hranice intervalů pro daný parametr. Tyto hranice jsou pak uloženy jako matice *nims* a *maxs*.

```
>> hP
MPE_P handle
    Properties:
    names: {'T'  'b'  'c'  'dx0'  'k'}
    mins: [1 1 1 0 1]
    maxs: [2 2 10 0 2]
    Methods, Events, Superclasses
```

4.2.4 Simulace (MPE_S)

Testované kombinace parametrů a chyba (rozdíl mezi simulací a měřením) se ukládá do objektu MPE_S. Objekt obsahuje matici *p*, kde řádky představují hodnoty pro jednotlivé parametry a sloupce jsou jednotlivé kombinace parametrů a konstant, které byly odsimulovány a matici, *error* ve které jsou v jednom řádku společné chyby pro všechny simulované kombinace modelů a dat. Sloupec chyby *error* odpovídá sloupci parametrů *p*. V proměnné *statistical_method* je název metody, podle které se počítá chyba.

Výsledky simulace jsou uloženy v tabulce *simresult*, kde řádky představují data a sloupce modely. V buňce pro danou kombinaci dat a modelu se nachází proměnná *gain*, která představuje hodnotu, kterou se násobí chyba. Vhodnou volbou této veličiny můžeme prohledávané metodě přidělit větší váhu pro danou kombinaci dat a modelu. Dále buňka obsahuje chybu pro danou kombinaci, která má stejnou strukturu jako celková chyba. Jako poslední se zde nachází matice, složená z výstupních vektorů simulace, kde sloupce představují jednotlivé simulace a řádky hodnoty výstupu pro dané časové úseky. Vzorkovací frekvence je vždy shodná se vzorkovací frekvencí měřených dat, které slouží pro určení chyby.

```
>> hS
MPE_S handle
    Properties:
    p: [7x100 double]
    simresult: {3x3 cell}
    error: [1x100 double]
    statistical_method: 'MSE'
Methods, Events, Superclasses
>> hS.simresult{1,2}
    gain: 1
    error: [1x100 double]
    y: [101x100 double]
```

4.3 Preprocessing

Zásadní roli pro úspěšné nalezení parametrů má vhodná volba vstupního signálu a správně zvolený model soustavy. Záleží též na vhodné metodice měření. Nahrazení soustavy modelem může být různé. Například zanedbání některých parametrů může ovlivnit hledání odhadu parametrů. Příprava dat zase ovlivňuje dobu simulace.

4.3.1 Příprava dat

Pro správné naměření verifikačních dat na soustavě musí být zaručeno, že je soustava izolována. Výstupní signál nesmí ovlivnit vstupní signál.

Vhodný vstupní signál je takový, který se na soustavě může objevit během provozu. Pokud se určitá dynamika soustavy neprojeví během takového signálu, tak se neprojeví ani během provozu a není motivace ji zjišťovat, pokud takový signál pro své vlastnosti nedovoluje provést identifikační experiment, například malá amplituda, nevhodné frekvenční spektrum nebo obtížné získání signálu. Může se použít uměle vytvořený signál. Takový signál lze jednoduše opakovaně generovat, matematicky popsat, fyzikálně realizovat a dostatečně vybudit systém vzhledem k jeho dynamice.[3][4].

Vstupní signál musí být navržen tak, aby nedošlo k saturaci výstupního signálu. Existují případy, kdy je saturace výstupního signálu žádoucí. Saturace výstupního signálu ze soustavy je nezbytná, pokud jsou hranice saturace součástí modelu a hodnota hranice saturace je součástí hledaných parametrů.

Dynamika soustavy se neprojeví ve všech naměřených signálech. Pokud bude změna vstupního signálu moc malá, neprojeví se dynamika systému. V takovém případě nedojde k nalezení parametrů, které souvisí s dynamikou systému, jako je hmotnost a moment setrvačnosti. Pokud bude vstupem rampa, tedy pomalý nárůst vstupní hodnoty, neprojeví se dynamika systému. Vhodnou volbou je tedy takový vstupní signál, který obsahuje ostrou změnu. Takovým signálem může být obdélníkový signál, kroková změna a další.

Pro hledání parametrů není vhodné, aby se signál periodicky opakoval. Takový signál by nepřinesl nové informace o soustavě, prodlužuje délku měření a tedy i dobu simulace. Vhodným kompromisem je použít periodický vstupní signál a výstupní signál zprůměrovat na jednu periodu. Tímto způsobem se částečně odstraní zašumění výstupního signálu. Je potřeba zajistit, aby v místě řezu periody byla počáteční a koncová hodnota stejná, včetně jejich derivací. Toho je většinou těžké dosáhnout. Lze toho dosáhnout, pokud se do vstupního signálu přidá ustálená hodnota, která bude vložena mezi původní periody. Tento vnořený signál musí mít takovou délku, aby se docílilo ustáleného stavu soustavy.

Pokud existuje naměřený signál soustavy, u kterého je vzorkovací frekvence příliš jemná, tedy pokud by snížení vzorkovací frekvence nevedlo k ztrátě informace v signálu obsaženém, je vhodné provést převzorkování, tedy nahrazení původní vzorkovací frekvenci novou, zpravidla menší. Pro převzorkování se využívá interpolace. Pro potřeby převzorkování lze použít funkci `hD.set_D`, v programu MPE popsanou v kapitole A.1.

Pro naměřený signál je možné použít některou z metod filtrace signálu. Tato práce nepopisuje metody filtrace, ani nástroj MPE neprovádí filtraci signálu. Je čistě na uživateli, aby využil vhodný matematický aparát na filtraci signálu. Šum, který působí na vstupní signál, nekazí odhad, zatímco šum, který působí na výstupní signál, odhad kazí.[1]

Parametry systému jsou také počáteční hodnoty integrálu, tedy počáteční stavy soustavy. Tyto parametry lze taky zahrnout do hledaných parametrů. To rozšíří hledaný prostor a počet

dimenzí, který je roven počtu těchto počátečních stavů. Ve většině případů lze buď zajistit nulové počáteční stavy nebo je naměřit. Pokud je tedy například u DC motoru vstupem napětí a výstupem natočení, je vhodné, aby signál napětí začínal nulovou hodnotou a nulovou směrnicí. Tímto přístupem lze určit počáteční stavy soustavy a snížit dimenzi hledaného prostoru.

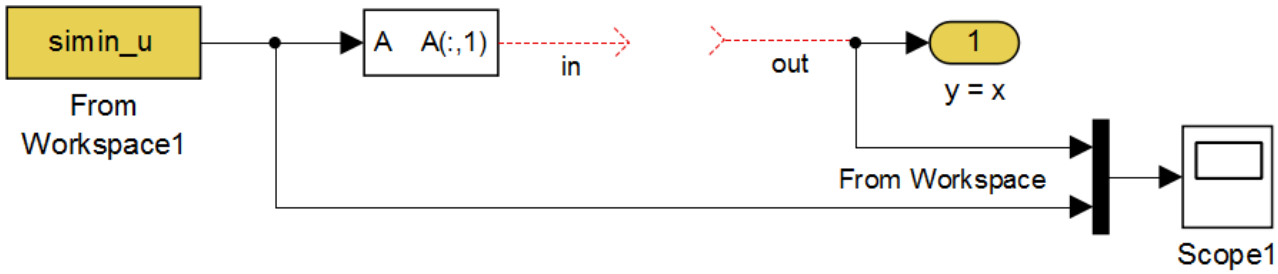
Program MPE vyžaduje, aby vzorkovací frekvence vstupního, výstupního a časového signálu byla stejná, tedy stejný počet vzorků.

4.3.2 Příprava modelu

Model nahrazuje soustavu matematickým popisem s přesností podstatnou pro vyřešení daného problému.[7] V této práci se pojmem model myslí náhrada soustavy modelem v prostředí Simulink. Model lze získat slovním popisem, fyzikálními zákony nebo identifikací systému z naměřených dat.

Důležitou otázkou je, kolik má systém, potažmo rovnice, zjistitelných parametrů. Pokud se popíše kyvadlo rovnicí 4.1 a za předpokladu, že z naměřených dat se získají $\varphi, \dot{\varphi}, \ddot{\varphi}$, lze získat jednoznačně jen dva parametry. Ze získaných parametrů p_1 a p_2 je možné získat jen parametr L, protože gravitace se bere jako známá konstanta. Pro parametry b a m je znám jen poměr mezi nimi, tedy je potřeba určit jeden z parametrů pro stanovení druhého.

Pro vytvoření nového modelu (nebo úpravu existujícího) poslouží model *new_model* ve složce */models*. Model obsahuje bloky potřebné pro program MPE. Na obrázku 4.2 lze vidět, kam se připojí model.



Obrázek 4.2: Prázdný model

Vytvořený program MPE umí nalézt parametry a konstanty obsažené v modelu, pokud se nachází v bloku *Constant* nebo *Gain*. Parametr může být součástí matematického vzorce nebo být obsažen vícekrát v modelu. Jsou bloky, které mohou také obsahovat hledaný parametr, jako jsou *Integrator* nebo *Saturation*. V těchto blocích lze přepnout zdroj tohoto parametru jako venkovní.

Parametry v modelu mohou mít libovolný název, který Simulink dovoluje. Může obsahovat číslici, ale nesmí číslicí začínat. Program rozlišuje velká a malá písmena (case-sensitiv). Důležitá pro dobu simulace a její přesnost je volba řešiče.

Program MPE je vyvíjen a testován ve verzi Matlabu r2012b. V případě komplikací s kompatibilitou je vhodné uložit model do této verze.

4.4 Přínosy MPE

Z hodnocení programu PE v kapitole 2.5 a z návrhů vylepšení v kapitole 3 byly stanoveny priority funkcionality nově vytvořeného nástroje MPE.

Vytvořený nástroj MPE je možné používat v Matlabu od verze 2012b v operačním systému Windows a OS X. Nástroj nevyžaduje přítomnost žádného dodatečného balíčku kromě Simulinku. Na rozdíl od programu PE, nový program MPE umožňuje spuštění ve studenské verzi Matlabu, která instalaci těchto balíčků kromě Simulinku neumožňuje. Některé prohledávací metody vyžadují některý z dodatečných balíčků, jako například *GA_Matlab*, který je obsažen v *Global Optimization Toolbox*.

Program MPE dává možnost vytvářet nové prohledávací metody bez nutnosti hluboké znalosti programu MPE. Podrobný návod přidání nové prohledávací metody je v kapitole A.3. Cílem bylo vytvořit takový nástroj, který může posloužit pro výuku předmětu, který se zabývá studiem prohledávacích metod. Součástí tohoto předmětu by mohlo být vytvoření prohledávací metody do programu MPE. Takto vytvořené algoritmy by bylo možné porovnat a zhodnotit jejich vlastnosti.

Přínosem je také uživatelské rozhraní programu MPE, které je více intuitivní než GUI PE. Program má česky psaný uživatelský manuál v kapitole A a také ukázkové studie v kapitole 4.6, které mohou posloužit studentům k pochopení práce s programem.

Běžné hledání parametrů může trvat i několik desítek hodin. Program MPE na rozdíl od programu PE umožňuje spojovat provedené hledání a tím čas rozdělit na více úseků. Program také poskytuje informaci o době, která zbývá do konce hledání.

Program MPE si ukládá průběhy všech simulací. Neztrácí žádné informace o prohledávaném prostoru, což umožňuje upravovat hledací prostor bez ztráty informace. Takový přístup k výsledkům simulací dovoluje změnu vyhodnocovací metody. Metody určovaní shody simulace a naměřených dat jsou v kapitole 4.4.5.

Výhodou je také možnost úplného prohledání prostoru, popsané v kapitole 4.5.1. Takové hledání je sice časově náročné, ale poskytuje důvěru v nalezené parametry.

Využitím tolerance při hledaní parametrů, popsané v kapitole A.3, se omezí počet simulací. Přidání tolerance zajišťuje, že se nebude simulovat kombinace parametrů, která je s určitou mírou tolerance stejná jako už dříve simulovaná kombinace parametrů. Taková kombinace parametrů by nepřinesla nové informace o prohledávaném prostoru. Tolerance lze v průběhu iteračního přístupu hledání měnit a ovlivnit tak dobu hledání. Iterační přístup hledání je popsán v kapitole 4.4.4.

Zásadní přínos do problematiky hledání parametrů přináší program MPE schopností kombinovat několik modelů a datových množin. Takovéto kombinace mohou sdílet některé parametry. Tímto definováním můžeme kombinacím přidat váhu, která představuje míru důvěry v danou kombinaci modelu a datové množiny.

4.4.1 Jeden model s více daty

Stejně jako program PE umí MPE pracovat s jedním modelem a více soubory naměřených dat.

Pro danou kombinaci parametrů se uskuteční simulace jednoho modelu se všemi datovými soubory a provede se ohodnocení jednotlivých kombinací. Na výsledné chyby se aplikují váhy a provede se výpočet společné chyby. Tuto chybu využije prohledávací metoda pro nalezení nové kombinace parametrů.

Kombinace jednoho modelu s více daty se využívá, pokud provedeme více měření s odlišným

vstupním signálem. Může také být rozdílná míra vybuzení, délka měření nebo míra změny.

Program umožňuje porovnat chybu stejné kombinace parametrů při použití různých datových souborů. Toto porovnání se provádí příkazem `hS.comparison_by_summed_error` a je popsáno v kapitole A.1.

4.4.2 Test různých variant modelů

Možnost, kterou program PE neumožňuje, je hledání parametrů pomocí více modelů. Program PE je vázán na jeden model. Nový program MPE není vázán na jeden model a dovoluje pracovat s libovolným počtem modelů. V rámci jednoho hodnocení se provede libovolná kombinace souborů dat a modelů.

Při tvorbě popisu soustavy modelem se často dospěje k více variantám modelů. Může to být způsobeno nedostatečnou znalostí soustavy nebo zanedbáním některých parametrů. V takovém případě je vhodné mít nástroj, který dokáže tyto modely porovnat. Takovýmto nástrojem je právě nový program MPE. Program dokáže porovnat, který model lépe sedí na určitý datový soubor.

Ohodnocení navržené kombinace parametrů se vypočítá váženým průměrem z jednotlivých kombinací souborů dat a modelů. Ohodnocení jednotlivých kombinací se uchovává pro případnou pozdější změnu vah.

Do ukázkových studií v kapitole 4.6 byly vybrány experimenty, které kombinují více modelů a souborů dat.

4.4.3 Několik různě provedených experimentů, které sdílí parametry

Pokud je možné vytvořit více různých variant experimentů, při kterých se některý z parametrů neprojeví, dosáhneme snížení dimenze prohledávacího prostoru. Snížení dimenze jen o jediný rozměr má výrazný vliv na schopnost nalézt parametry modelu.

Parametry jsou sdílené v rámci modelů, i když nejsou ve všech modelech obsaženy všechny. Každý model může mít tedy svoje unikátní a sdílené parametry. Příkladem může být oscilátor z kapitoly 4.6.1, kde unikátní parametr je koeficient tření.

Pokud máme více různě provedených experimentů, které sdílejí parametry, pak hledáme takové hodnoty parametrů, které nejlépe sedí na všechny modely. Přidáním váhy jednotlivým kombinacím modelů a souborů dat se ovlivní prohledávací metoda.

Celá problematika se týká nalezení takového experimentu, u kterého buď lze zanedbat určitý parametr nebo změnit hodnotu parametru o známou hodnotu. Změna hodnoty parametru nesmí ovlivnit jiný parametr. Takovým parametrem může být hmotnost nebo moment setrvačnosti, které lze změnit o přesně danou hodnotu. Zároveň výrazně ovlivní dynamiku systému. Takováto změna parametrů vyžaduje dobrou znalost soustavy.

Jako příklad sdílení parametrů poslouží kyvadlo, tedy dynamický systém druhého řádu. Jak bylo popsáno v kapitole 4.3.2, nelze z rovnice 4.1 pomocí měření získat všechny parametry. Pro parametry m a b nelze určit jejich hodnoty, pouze jejich poměr.

$$mL^2\ddot{\varphi} = -mgL\sin(\varphi) - b\dot{\varphi} \tag{4.1}$$

Kde $m[kg]$ je hmotnost, $L[m]$ délka od osy rotace po těžiště hmoty, $b[s^-1]$ součinitel útlumu,

$\varphi[rad]$ natočení kyvadla.

$$\ddot{\varphi} = -p_1 \sin(\varphi) - p_2\dot{\varphi} \tag{4.2}$$

Kde p_1 a p_1 jsou:

$$p_1 = \frac{g}{L} \tag{4.3}$$

$$p_2 = \frac{b}{mL^2} \tag{4.4}$$

Pro zjištění chybějících parametrů b a m lze upravit experiment tak, aby se jeden z parametrů změnil o známou velikost. Tlumení se velice obtížně mění tak, aby byla známá jeho přidaná hodnota, proto je vhodnější upravit hmotnost. Pokud se vytvoří nový experiment, který bude mít přidanou definovanou zátěž, lze pomocí vzorce 4.8 a 4.9 stanovit chybějící parametry. Těžiště přidané a původní hmotnosti musí být stejné, aby parametr L zůstal stejný.

Pokud by se problém řešil v programu PE, naměřily by se dva soubory dat (s a bez přidané hmotnosti). Provedl by se první odhad parametrů, kde by se získaly hodnoty parametrů p_1, p_2. Ve druhém odhadu parametrů by se získala hodnota parametrů $\acute{p_2}$. Jak ukazuje ukázková studie DC motoru 4.6.3, nevede tento přístup k důvěryhodným hodnotám, protože program PE nehledá kompromis v chybě mezi prvním a druhým experimentem.

$$(m + m_z)L^2\ddot{\varphi} = -(m + m_z)gL \sin(\varphi) - b\dot{\varphi} \tag{4.5}$$

$$\ddot{\varphi} = -p_1 \sin(\varphi) - \acute{p_2}\dot{\varphi} \tag{4.6}$$

$$\acute{p_2} = \frac{b}{(m + m_z)L^2} \tag{4.7}$$

$$m = \frac{m_z\acute{p_2}}{p_2 - \acute{p_2}} \tag{4.8}$$

$$b = \acute{p_2}L^2(m + m_z) \tag{4.9}$$

Program MPE na rozdíl od programu PE umožňuje pracovat s více modely. Tato schopnost se využije tak, že se vytvoří modely pro rovnice 4.1 a 4.5. Program provádí hledání odhadu parametrů tak, aby společná chyba byla co nejmenší. Program tedy hledá určitý kompromis. Pro jednotlivé experimenty lze nastavit váhu, která ovlivní prohledávací metodu.

4.4.4 Iterační změna prohledávacího prostoru

Z kapitoly 4.5.1 plyne, že úplné prohledání prostoru není ve výpočetních silách. Přesto je hrubé prozkoumání prostoru velmi důležité. Poskytuje znalost prostoru, která poslouží pro nastavení intervalu některé inteligentní prohledávací metody nebo počáteční polohy pro gradientní metody, tedy vytvoření takzvaně lukrativních lokalit.

Program PE umožnuje iterační přístup takový, že lze některé z hodnot výsledných para-

metrů použít pro nové hledaní. Problém nastane, pokud se parametr nachází ve svém lokálním minimu, ze kterého se i po novém zahájení hledání nedostane. PE si také neukládá informace z dřívějšího hledání a tedy ztrácí informace o prohledávacím prostoru.

Nový program MPE po zakončení hledání (odsimulování zadaného počtu simulací) vizualizuje dosaženou chybu na osách jednotlivých intervalů. Uživatel může buď omezit interval pro nové hledání nebo zahájit hledání gradientní metodou. Gradientní metoda využije kombinaci parametrů s nejmenší chybou jako počáteční polohu. Lze si to představit jako kapky deště, které si najdou cestu do údolí z mraků, které jsou pod vrcholky hor, tedy omezení plochy z které prší do údolí. Podrobný návod práce s iteračním přístupem v programu MPE je v kapitole A.2.

4.4.5 Metody určování shody simulace a naměřených dat

Volba metody určování chyby, neboli ohodnocování, má vliv na schopnost nalézt hledané parametry. Existuje více metod výpočtu chyby mezi měřením a simulací. V programu PE se využívají dvě metody, SSE (The sum of squares due to error) a SAE (Sum absolute error), popsané v kapitole 2.1.[2]

Každá kombinace souboru dat a modelu má pro každou kombinaci parametrů vypočítanou chybu. Dále existuje globální chyba, která se počítá jako průměr chyb jednotlivých kombinací souborů dat a modelů. Podle této společné chyby se prohledávací metoda rozhoduje o nových kombinacích parametrů, tedy o poloze v prohledávacím prostoru.

Průměrná kvadratická chyba MSE

Průměrná kvadratická chyba zohledňuje vzdálenost mezi každou hodnotou v průběhu simulace a měření. O metodě se také mluví jako o momentu setrvačnosti chyby. Jedná se o nejrozšířenější metodu výpočtu chyby. Využívá se hlavně ve statistice. Metodu představil C.F.Gauss.[10]

Výpočet chyby metodou MSE je náchylný na extrémní výchylky v signálu, které mohou být způsobené chybou v měření. Takový soubor je potřeba filtrovat, což může vést ke ztrátě informace obsažené v signálu.[9]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 \tag{4.10}$$

V rovnici 4.10 je y naměřený signál na soustavě a y' signál vytvořený simulací.

Průměrná absolutní chyba MAE

Průměrná absolutní chyba poskytuje stejnou váhu všem chybám. Je méně náchylná na extrémní výchylky v signálu.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'| \tag{4.11}$$

V rovnici 4.11 je y naměřený signál na soustavě a y' signál vytvořený simulací.

Odmocněná průměrná kvadratická chyba RMSE

Odmocněná průměrná kvadratická chyba znevýhodňuje rozptyl, protože dává chybám s větší absolutní hodnotou větší váhu, než chybám s menší absolutních hodnotou. RMSE samozřejmě

nikdy nebude menší než MAE. RMSE má tendenci nabývat větších hodnot než MAE, protože její spodní limit je MAE a horní limit $MAE \times \sqrt{n}$.

$$RMSE = \sqrt{MSE} \tag{4.12}$$

V rovnici 4.12 je MSE průměrná kvadratická chyba z rovnice 4.10.

Normovaná odmocněná průměrná kvadratická chyba NRMSE

Normovaná odmocněná průměrná kvadratická chyba je užitečná, protože je snaha srovnávat RMSE s různými jednotkami.

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \tag{4.13}$$

V rovnici 4.13 je $RMSE$ průměrná kvadratická chyba z rovnice 4.12, y_{max} a y_{min} je maximální a minimální hodnota obsažená v naměřeném signálu na soustavě.

Špičkový poměr signálu k šumu (PSNR)

Špičkový poměr signálu k šumu vyjadřuje poměr mezi maximální možnou energií signálu a energií šumu.[2]

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right) \tag{4.14}$$

$$R = max(y_{max}, y'_{max}) \tag{4.15}$$

V rovnici 4.14 je R maximální hodnota, která se nachází v naměřeném nebo odsimulovaném signálu.[2]

4.5 Metody prohledávání

4.5.1 Grid

Metoda je věrná svému jménu, je založena na rozdělení zadaného intervalu na přesný počet hodnot. Pokud by se tedy odhadoval parametr, který je zadán uzavřeným intervalem <1;5> s rozdělením na tři hodnoty, interval by se rozdělil na N-1 intervalů, kde N představuje hodnotu gridu a hranice mezi intervaly jsou hledané hodnoty. Simulace by se tedy uskutečnila pro hodnoty 1, 3, 5, pokud bychom měli interval z jedné strany otevřený <1;6). (Do intervalu spadají všechny hodnoty, kromě hodnoty z otevřené strany.) Interval bychom rozdělili na N intervalů, kde hledané hodnoty jsou hranice mezi intervaly bez hodnoty, která leží na otevřené straně intervalu. Tedy 2,4,6 pro rozdělení na tři hodnoty. Pokud by byl interval otevřený (1;7), rozdělil by se na N+1 intervalů a hledané hodnoty by byly hranice mezi intervaly, kromě hodnot na hranici původního intervalu.

Metoda je vhodná, pokud je obava uvíznutí jiné metody v lokálním minimu. Tento přístup nevyužívá žádné znalosti z předchozích simulací, které jsou v jiných metodách použity pro zrychlení nalezení hledaných parametrů. Počet simulací je roven N^P, kde N je míra rozdělení a

Tabulka 4.1: Tabulka parametrů pro metodu Grid

Parametr metody	**Název parametru v metodě**	**Výchozí hodnota**
Maximální počet simulací	*NumberOfSimulations*	400
Zobrazit průběh	*ShowSimulation*	Off
Tolerance shody	*Tolerance*	0

P je počet parametrů. Při běžně velkém počtu parametrů je potřeba mnohem větší počet simulací, než při inteligentním prohledávaní, ale není potřeba provádět výpočet po každé simulaci či generaci. Prohledáním celé oblasti dostaneme informace o prohledávaném prostoru, metoda může tedy sloužit pro určení počátečního intervalu pro některou inteligentní metodu.

Metoda má v programu MPE jenom při parametry *NumberOfSimulations*, *ShowSimulation* a *Tolerance*. Parametr *NumberOfSimulations* určuje maximální počet simulací. Metoda najde největší počet permutací s opakovaním, který nepřesáhne zadaný počet simulací. Zvolená hodnota *Tolerance* určuje jakou vzdálenost od odsimulovaných kombinací parametrů považujeme za nepotřebnou pro novou kombinaci parametrů. Pokud se nezmění žádná z hranic intervalů pro hledání, tak i při malé hodnotě tolerance nedojde k simulování nových hodnot. Důvodem je shoda navržených kombinací parametrů s kombinacemi, které byly odsimulovány v předchozím použití této metody. Pokud je tedy tolerance aplikována, je vhodné pro každou novou aplikaci této metody změnit hranice intervalu nebo počet simulací.

Výhodou této metody je také možnost vizualizace prostoru pro dva parametry. Ostatním parametrům se určí hodnota pro tuto vizualizaci. Vytvoří se terén, kde údolí představují lokální minima. Na obrázku 4.3 je použita metoda Grid. Jedná se o oscilátor z kapitoly 4.6.1.

Pro vizualizaci na obrázku 4.3 bylo potřeba 100^2 simulací, které trvaly tři a půl minuty. Pokud by se aplikovala metoda pro všech sedm parametrů, jednalo by se o 100^7 simulací. Simulace by trvala 66590 let. Pokud by se ukládaly informace o simulaci stejně jako při použití programu MPE, bylo by potřeba 71525 TB. Z těchto výpočtů vyplývá, že prohledání celého více-dimenzionálního prostoru s rozumnou přesností není možné. Metoda se tedy využívá jako první hrubé prozkoumání prostoru, které vytvoří vhodné počáteční polohy pro jiné metody. Parametry pro tuto metodu jsou v tabulce 4.1.

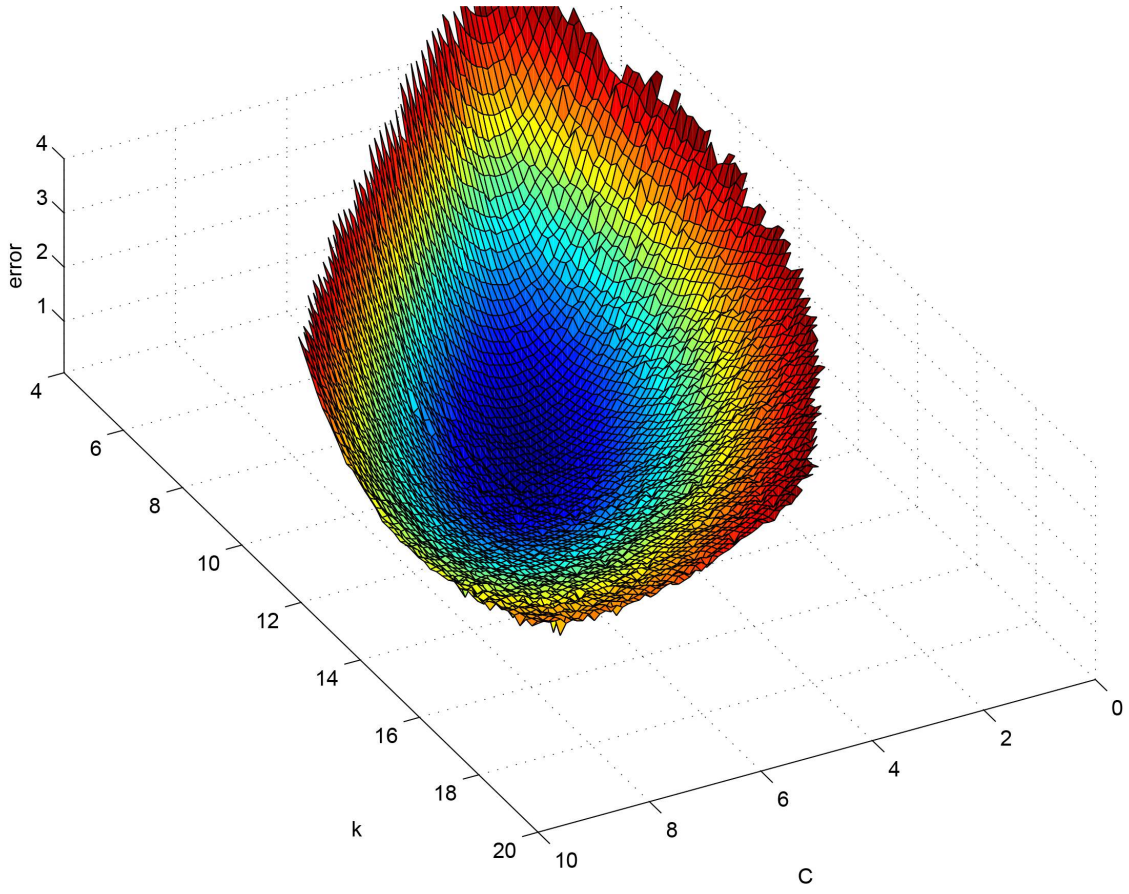
Pokud se některé z lokálních minim nachází na hranici intervalu, je vhodné interval rozšířit tímto směrem. Tento přístup může posloužit jako základ gradientní metody pro získání směru nejrychlejšího spádu.[6]

4.5.2 Monte Carlo

Monte Carlo je stochastická metoda, založená na náhodném zastoupení prostoru tak, aby rozložení bylo rovnoměrné. Metoda patří do neinformovaných metod, protože pro vygenerování nových hodnot nevyužívá znalost získanou z dřívějšího hledání. Výhodou metody je nezávislost vygenerovaných hodnot na ohodnocení, tedy netrpí uvíznutím v lokálním extrému a je jednoduše implementovatelná. Nevýhodou je obvykle větší počet simulací potřebných pro nalezení globálního minima než u informovaných metod.

Metoda je závislá na generátoru náhodných čísel, respektive pseudonáhodných čísel, tedy aby rozložení vygenerovaných čísel bylo rovnoměrné. Program Matlab využívá pro generování náhodných čísel řadu, která zajišťuje rovnoměrné rozložení vygenerované matice čísel.[2]

Jako první zmínka o metodě Monte Carlo se často uvádí Boffonova úloha, která z náhodného



Obrázek 4.3: Grid pro dva parametry c a k

umístění dopadnutých jehel dokáže určit hodnotu pí. Významnějšího využití se metoda dočkala až s rozvojem výpočetní techniky. Konkrétně ve druhé světové válce, kde našla uplatnění při vývoji atomové zbraně v projektu Manhattan.[5]

Metoda je implementována v programu MPE pod názvem *MC* a možné nastavení algoritmu je v tabulce 4.2.

4.5.3 Genetický algoritmus

Genetický algoritmus vychází ze znalosti evoluce. Poprvé byl algoritmus představen Johnem Hollandem v roce 1975 v publikaci *Adaptation in Natural and Artificial Systems* [8]. Algoritmus napodobuje schopnost přirozeného výběru silnějších jedinců.

Důležité pojmy pro genetický algoritmus jsou: jedinec, generace, elita, mutace, hodnotící

Tabulka 4.2: Tabulka parametrů pro metodu Monte Carlo

Parametr metody	**Název parametru v metodě**	**Výchozí hodnota**
Počet simulací	*NumberOfSimulations*	100
Zobrazit průběh	*ShowSimulation*	Off
Tolerance shody	*Tolerance*	0

Tabulka 4.3: Tabulka parametrů pro Genetický algoritmus

Parametr metody	**Název parametru v metodě**	**Výchozí hodnota**
Počet generací	*Mutation*	0.03
Zobrazit průběh	*ShowSimulation*	Off
Tolerance shody	*Tolerance*	0
Časový limit	*TimeLimit*	inf
Počet jedinců v elitě	*EliteCount*	5
Část generace určená pro křížení	*CrossoverFraction*	0.8
Počet generací	*Generations*	10
Počet jedinců v generaci	*PopulationSize*	50

funkce, selekce a křížení. V programu MPE je navržená kombinace parametrů jedincem a generace je skupina kombinací určená pro hledání odhadu parametrů. Elita je určitý počet kombinací parametrů, které byly ohodnoceny nejmenší chybou. Mutace je změna hodnoty daného parametru. Hodnotící funkce je porovnání odezvy simulace pro danou kombinaci parametrů s odezvou soustavy. Selekce je výběr nejlepších kombinací parametrů pro další generaci. Křížení je prohození některých parametru mezi jedinci.[8]

V programu MPE jsou dva genetické algoritmy, jeden byl vytvořen v rámci jiné práce a druhý upravuje GA obsažení v Matlabu. Možné nastavení algoritmu je v tabulce 4.3. K podrobnějšímu nastavení je potřeba upravit metodu *GA_Matlab*.

Doporučené nastavení pro velikost generace je desetkrát větší než počet hledaných parametrů. Míra mutace by neměla překročit pět procent.[8]

4.5.4 Gradientní metody

Program MPE využívá gradientní metody obsažené ve funkci *fsolve* k nalezení lokálního minima, kde počáteční polohy jsou kombinace parametrů s nejlepším ohodnocením. Metody jsou založeny na znalosti největšího spádu. Program MPE má jako výchozí metodu Levenberg – Marquardt. Všechny metody obsažené v programu:

- Levenberg – Marquardt
- Trust region reflective
- Trust region dogleg

Trust region reflective je algoritmus, který je účinný na řídké problémy. Je možné použít speciální techniky, jako například Jacobianovu vícenásobnou funkci pro rozsáhlé problémy.[2]

Trust region dogleg je jediný algoritmus, který je speciálně určen na řešení nelineárních rovnic. Ostatní metody minimalizují součet čtverců funkce.[2]

Program MPE provede hledání ze všech vybraných kombinací, proto je potřeba omezit tento počet kombinací zmenšením intervalu chyby. Hledání pokračuje do té doby, než je nalezeno lokální minimum. Doba hledání se tedy mění podle vzdálenosti a spádu počáteční polohy od minima.

4.6 Ukázkové studie odhadu parametrů

4.6.1 Oscilátor

Jako ukázková úloha je použit mechanický oscilátor, tedy kmitající hmota s pružinou a tlumičem. Pro demonstraci schopností programu MPE jsou vytvořené dva experimenty, které sdílejí parametry. Změnou experimentu ovlivníme rovnici popisující soustavu, ale neovlivníme její parametry. Odebraný parametr z naměřených dat nelze získat, ale snížením dimenze prohledávaného prostoru umožníme snáze najít zbylé parametry.

Za předpokladu nulové počáteční polohy a zrychlení se hledají hodnoty pěti parametrů, v obou experimentech se projeví čtyři. Pokud by experiment nezačínal nulovou polohou a rychlostí, pak se hodnoty těchto stavů daly určit z naměřených dat.

$$m\ddot{x} = -kx - b\dot{x} - Tsgn(\dot{x}) + mg + cF \tag{4.16}$$

V rovnici 4.16 je m hmotnost, k tuhost, b tlumení, T třecí síla, g gravitační zrychlení, x poloha, $\dot{x}$ rychlost a $\ddot{x}$ zrychlení. T je třecí síla, nikoliv koeficient tření, který působí vždy proti směru pohybu.

Pokud se provedou dva experimenty o čtyrech parametrech místo jednoho experimentu o pěti parametrech, sníží se výrazně hledaný prostor. Pokud by se prostor prohledával metodou Grid o počtu deseti vzorků, pro jeden model o pěti parametrech by se jednalo o 100 000 simulací. Pokud by se provedlo hledání parametrů na modelu o čtyřech parametrech, bylo by to 10 000 simulací. Po nalezení čtyř sdílených parametrů by se provedlo hledání na modelu o jednom neznámém parametru, tedy 10 simulací. Dohromady tedy 10 010 simulací oproti 100 000 simulací o stejné vypovídající hodnotě. Pokud by se provedlo hledání na obou modelech o 50 000 simulacích naráz, tedy 100 000 dohromady, dal by se jeden parametr rozdělit na 15 hodnot místo 10.

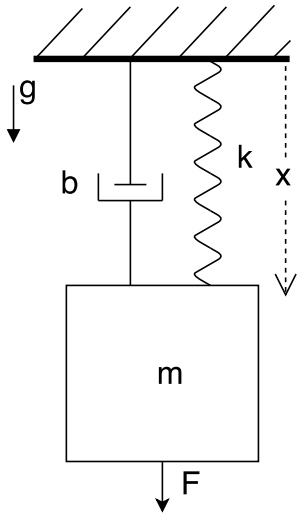
První experiment s vlivem gravitace a bez tření

První experiment, znázorněný na obrázku 4.4, je bez vlivu tření. Soustava je popsána rovnicí 4.17. Hmota je zavěšená na pružině s tlumičem a kmitá svisle v ose gravitačního zrychlení. Oscilátor je vybuzen silou cF ve směru gravitačního zrychlení.

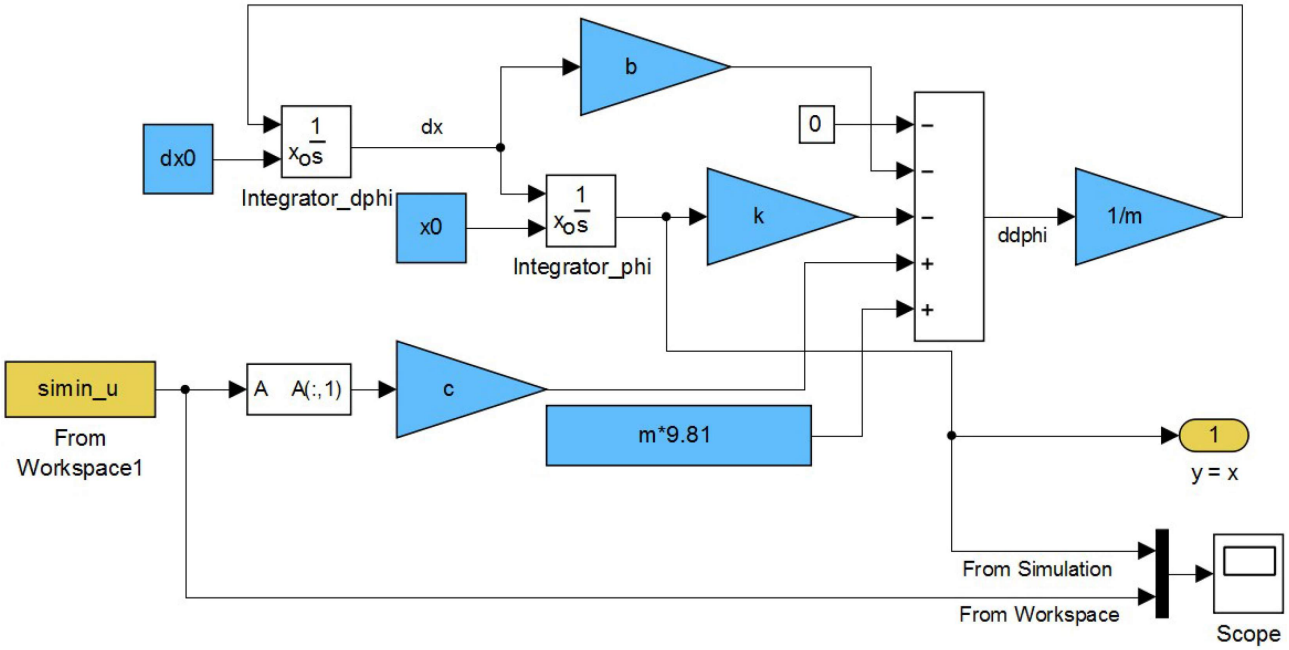
$$m\ddot{x} = -kx - b\dot{x} + mg + cF \tag{4.17}$$

V rovnici 4.17 je m hmotnost, k tuhost, b tlumení, g gravitační zrychlení, x poloha, $\dot{x}$ rychlost a $\ddot{x}$ zrychlení.

Model pro první experiment je znázorněn na obrázku 4.5. Vstupní síla pro první experiment je na obrázku 5.14. Naměřená poloha je znázorněna na obrázku 5.15.



Obrázek 4.4: První experiment s vlivem gravitace a bez tření



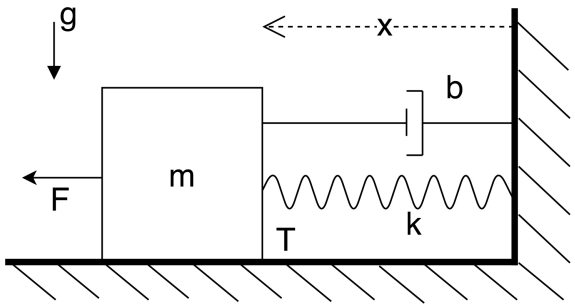
Obrázek 4.5: Model M4 prvního experimentu s vlivem gravitace a bez tření

Druhý experiment bez vlivu gravitace a se třením

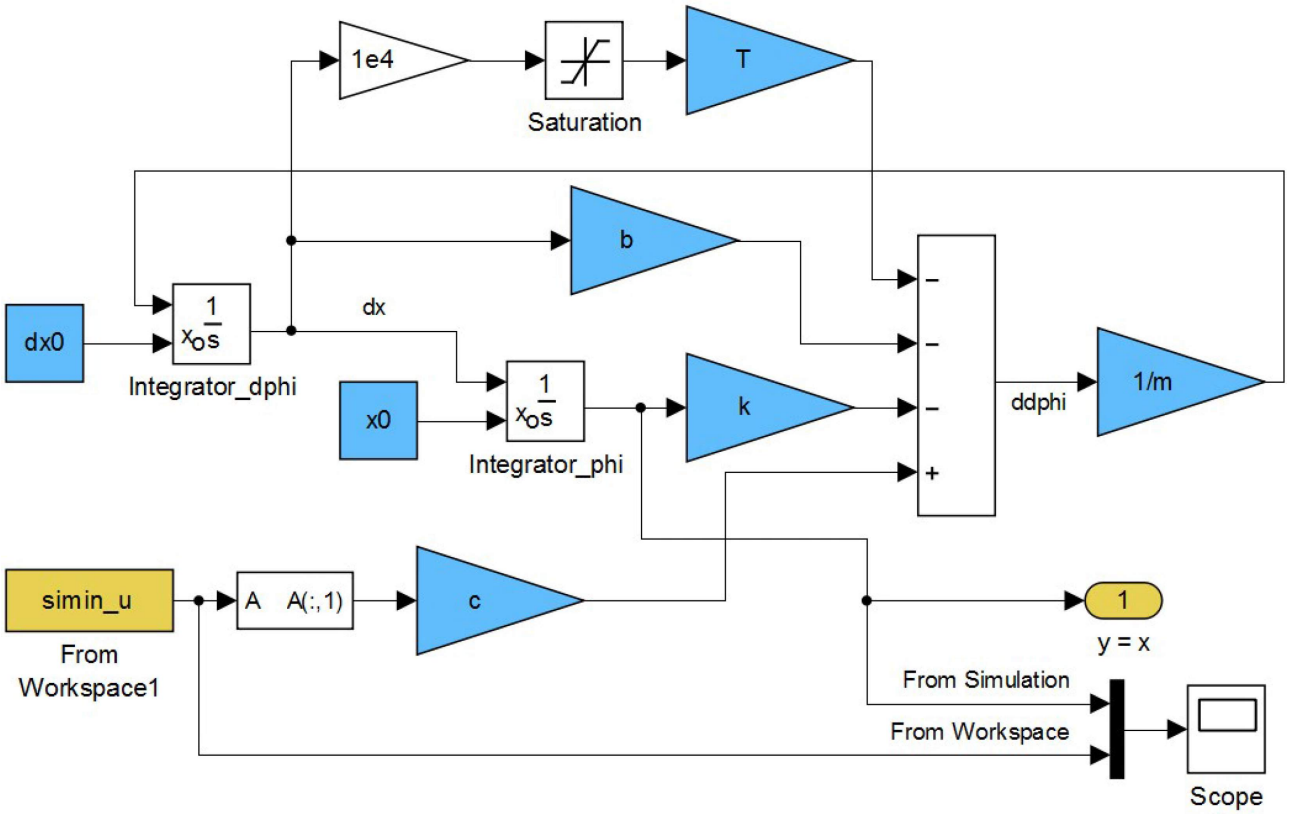
Druhý experiment, znázorněný na obrázku 4.6., je bez vlivu gravitační síly. Jak už bylo napsáno, T je třecí síla, která se skládá z normálové síly, na kterou má vliv gravitační zrychlení, přesto se tento experiment popisuje jako experiment bez vlivu gravitace. Soustava je popsána rovnicí 4.18. Předmět se pohybuje po vodorovné ploše, která na něj působí silou T proti směru pohybu. Je vybuzován vodorovnou silou cF.

$$m\ddot{x} = -kx - b\dot{x} - Tsgn(\dot{x}) + cF \tag{4.18}$$

V rovnici 4.18 je m hmotnost, k tuhost, b tlumení, g gravitační zrychlení, T třecí síla, x poloha, $\dot{x}$ rychlost a $\ddot{x}$ zrychlení.



Obrázek 4.6: Druhý experiment bez vlivu gravitace a se třením



Obrázek 4.7: Model M3 prvního experimentu bez vlivu gravitace a se třením

Model pro druhý experiment je znázorněn na obrázku 4.7. Vstupní síla pro druhý experiment je na obrázku 5.16. Naměřená poloha je znázorněna na obrázku 5.17.

Definice problému

Protože se jedná o ukázkovou studii, která byla vytvořena simulací, neexistuje tedy reálná soustava, ale jsou známé přesné parametry. V takovém případě můžeme provést srovnání a určit míru shody s odhadnutými parametry.

Je proveden odhad pomocí tří metod: Gridu, Monte Carlo a Genetického algoritmu. Tedy pomocí metod, které byly vytvořeny v rámci této práce. Program obsahuje více metod, které ale jsou vytvořeny v rámci jiné práce a proto jejich zhodnocení nebude obsahem této práce.

Bylo odsimulováno 100 000 simulací pro každý experiment, tedy 200 000 simulací v rámci jednoho odhadu parametrů. Interval parametrů byl zvolen tak, aby spodní hranice byla nulová

hodnota a horní hranice dvojnásobek předpokládané hodnoty. Tento interval byl také zvolen tak, aby hledaná hodnota byla přesně mezi dvěma vzorky pro metodu Grid, tedy aby se mřížka metody Grid neprotínala s hodnotami hledaných parametrů. Pro hmotnost byl spodní interval posunut od nuly, aby nedošlo k dělení nulou, ke kterému by v případě použití metody Grid došlo určitě a v případě použití ostatních metod by k němu mohlo teoreticky dojít také. Taková nedovolená operace by zapříčinila konec chodu programu.

V tabulce 4.4 je tučně znázorněna kombinace souborů dat a modelů, které odpovídají popsaným experimentům. Pro doplnění tabulky byly vytvořeny modely a soubory dat i pro zbylé kombinace. Pro hledání parametrů jsou použity kombinace D1,M2 a D4,M2.

Tabulka 4.4: Tabulka kombinací souborů dat a modelů pro oscilátor

Působí tíhová síla		Ano	Ne
Působí tření	Ano	D2,M1	**D1,M2**
	Ne	**D4,M4**	D3,M3

Podle tabulky 4.4 je definován problém pomocí příkazů uvedených v příloze B. Metodika stanovení chyby je metoda nejmenších čtverců (MSE). Hodnota váhy pro jednotlivé kombinace byla stanovena pokusem. Pokus se skládal z odsimulování 1000 simulací a váha pro jednotlivé kombinace se stanovila tak, aby průměrná chyba byla pro obě kombinace stejná. Takto zvolený přístup stanoví pro obě kombinace stejnou váhu důvěry. Váhy jsou také vynásobeny stejnou hodnotou, aby se výsledná chyba pohybovala v řádech, které program zobrazuje.

Definice problému pro program MPE pomocí příkazů je v příloze B. Nastavení pro genetický algoritmus je v tabulce 4.5, pro Grid v tabulce 4.6 a pro Monte Carlo v tabulce 4.7.

Tabulka 4.5: Tabulka nastavených parametrů pro Genetický algoritmus

Parametr metody	**Název parametru v metodě**	**Hodnota**
Počet simulací pro jeden model	-	1e5
Počet generací	*Generations*	50
Velikost populace	*PopulationSize*	2000
Míra mutace	*Mutation*	3%
Počet elitních jedinců	*EliteCount*	5

Tabulka 4.6: Tabulka nastavených parametrů pro metodu Grid

Parametr metody	**Název parametru v metodě**	**Hodnota**
Počet simulací pro jeden model	*NumberOfSimulations*	1e5
Počet vzorků na parametr	-	10

Tabulka 4.7: Tabulka nastavených parametrů pro metodu Monte Carlo

Parametr metody	**Název parametru v metodě**	**Hodnota**
Počet simulací pro jeden model	*NumberOfSimulations*	1e5

Nalezené parametry

V tabulce 4.8 jsou hodnoty parametrů, které mají nejmenší společnou chybu z jednotlivých prohledávacích metod. Protože hodnoty parametrů jsou v tomto případě známé, je možné porovnat odhadnuté parametry s parametry soustavy.

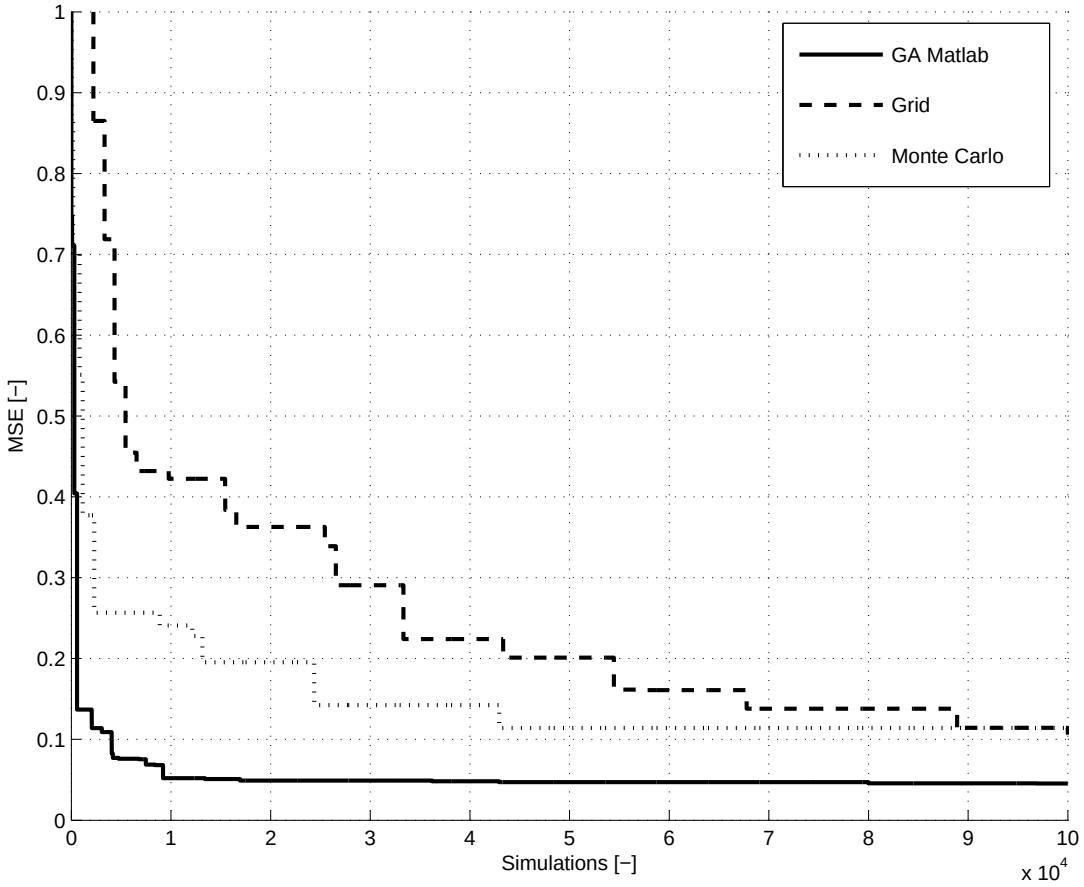
Tabulka 4.8: Tabulka odhadnutých parametrů

Parametry	**T**	**b**	**c**	**k**	**m**
Jednotky	[N]	[Ns/m]	[-]	[N/m]	[kg]
Hodnota parametru	0.1	0.9	5	10	0.25
Hranice intervalu	<0; 0.2>	<0; 1.8>	<0; 10>	<0; 20>	<0.01; 0.51>
Grid	0.111	0.800	4.444	8.889	0.232
Chyba	0.011	0.1	0.556	1.111	0.018
Procentuální chyba	11%	11%	11%	11%	7.2%
Monte Carlo	0.117	1.1845	5.866	11.797	0.289
Chyba	0.017	0.285	0.866	1.797	0.039
Procentuální chyba	17%	32%	17%	18%	16%
Genetický Algoritmus	0.129	0.917	4.973	9.986	0.251
Chyba	0.029	0.017	0.027	0.014	0.001
Procentuální chyba	29%	1.9%	0.54%	0.14%	0.4%

Z tabulky 4.8 vyplývá, že nejlépe dopadl genetický algoritmus. Metody Monte Carlo a Grid jsou ovlivněny zvoleným intervalem. V prohledávaném prostoru dokážou stanovit zajímavou oblast pro další hledání odhadu některou z gradientních metod.

Znázornění rychlosti nalezení hledaných parametrů je v grafu 4.8. Pro vytvoření tohoto grafu je použit níže napsaný cyklus, který určí, jaká minimální hodnota byla během simulace nalezena. Z grafu vyplývá, že Genetickému algoritmu stačí 10 000 simulací pro nalezení parametrů, které se v dalším prohledávaní zlepší už jen minimálně.

```
minim=inf;
for i=1:length(hS.error)
if hS.error(1,i)<minim
minim = hS.error(1,i);
end
progress(i)=minim;
end
```



Obrázek 4.8: Minimální chyba v průběhu hledání

Grafy 5.20 a 5.19 znázorňují odezvu modelu pro hodnoty nalezených parametrů, porovnanou s odezvou hledaných parametrů. Z grafů vyplývá, že odezvy se velice shodují a hledané minimum chyby je tedy velice mělké, tedy malá změna parametru vede k minimální změně chyby.

Zhodnocení odhadnutých parametrů

Po úpravě rovnice 4.16 na 4.19 vyplyne, že počet skutečně možných parametrů, které lze z naměřených dat odhadnout, je čtyři. Stále platí, že vhodnou volbou experimentu lze snížit počet parametrů.

$$\ddot{x} = -p_1 x - p_2 \dot{x} - p_3 sgn(\dot{x}) + g + p_4 F \tag{4.19}$$

Kde p_1, p_2, p_3 jsou:

$$p_1 = \frac{k}{m} \tag{4.20}$$

$$p_2 = \frac{b}{m} \tag{4.21}$$

$$p_3 = \frac{T}{m} \tag{4.22}$$

$$p_3 = \frac{c}{m} \tag{4.23}$$

Lze nalézt nekonečně velký počet kombinací parametrů m, k, b, T, c, které budou mít stejné hodnoty parametrů p_1, p_2, p_3. Tedy stejnou dynamiku a stejnou chybu. Je potřeba zdůraznit, že nalezené parametry nemusí být parametry soustavy. Pro jednoznačné určení parametrů m, k, b, T, c z parametrů p_1, p_2, p_3 je potřeba buď změřit některý z parametrů, nebo provést experiment, který některý z parametrů změní o známou hodnotu. Takový přístup je popsán v kapitole 4.6.3.

4.6.2 Škrtící klapka

Škrtící klapka je součástka, která otevírá nebo uzavírá průtok potrubím. Používá se v zážehových motorech, kde pomocí natočení klapky reguluje množství paliva proudícího do motoru. Soustava má významné suché tření. Klapka obsahuje předpjatou pružinu, která klapku v havarijním stavu uzavře.[11][12]

Soustava byla vybrána jako ukázková studie pro program MPE z důvodu významné nelinearity. Problematikou řízení škrtící klapky se zabývají práce [11][12]. Tato práce se bude zabývat pouze odhadem parametrů modelu, který reprezentuje soustavu. Škrtící klapka je na obrázku 4.9.

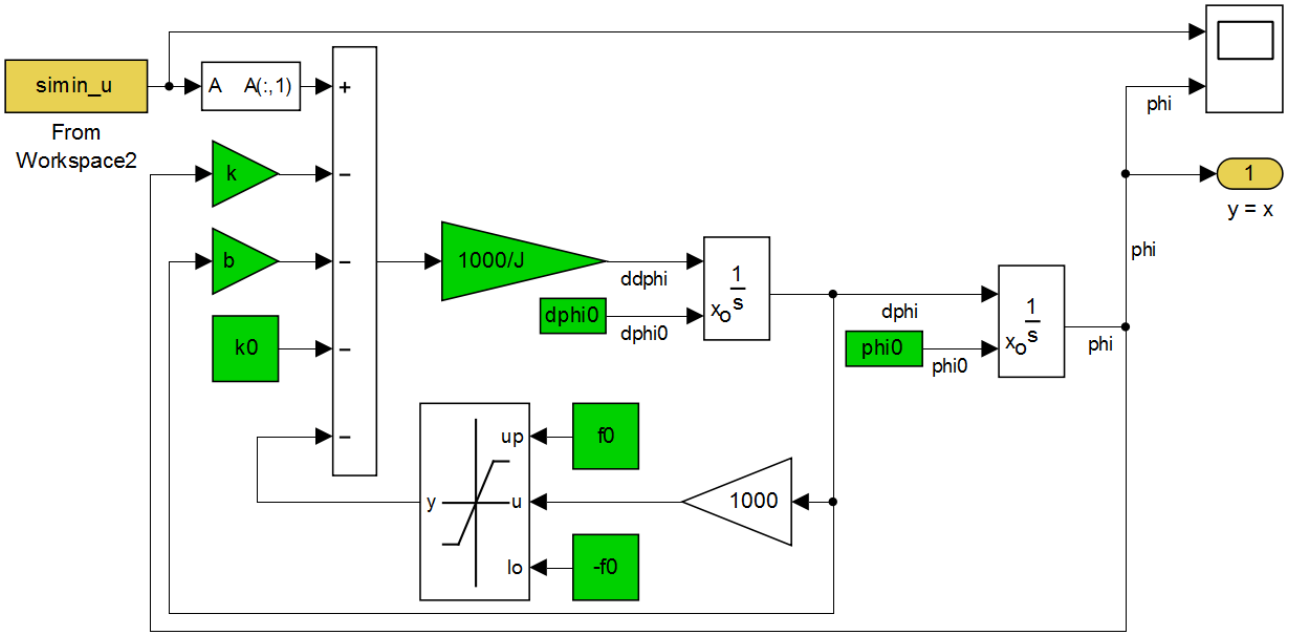
$$J\ddot{\varphi} = M - b\dot{\varphi} - k\varphi - f_0 sgn(\dot{\varphi}) - k_0 \tag{4.24}$$

Rovnici 4.24 by bylo nutné upravit stejným způsobem, jako rovnici 4.16 na 4.19, pokud by parametr f_0 nebyl znám. Protože parametr f_0 byl určen experimentem, není taková úprava zapotřebí.

Naměřená data a vytvořený model

Na škrtící klapce byly naměřeny série dat s různou dynamikou. Jeden ze vstupních signálů je v grafu 5.18. Pro každý soubor dat byly zjištěny z naměřených dat počáteční stavy $dphi0aphi0$. Pro každý soubor dat je jeden model, který má jiné počáteční stavy. Model pro škrtící klapku je na obrázku 4.10.

Obrázek 4.9: Elektronická škrticí klapka [12]



Obrázek 4.10: Model škrtící klapky

Definice problému

Interval pro hledaný parametr byl zvolen od nuly do dvojnásobku očekávané hodnoty parametru. Pro hledání odhadu parametrů se použily čtyři metody: Grid, Monte Carlo, Genetický algoritmus a Iterační přístup.

Postup hledání iteračním přístupem je v tabulce 4.9. Jde o zmenšování hledaného prostoru na základě získaných znalostí z dřívějších ohodnocení. Ve sloupci Δ je vypočítáno procentuální zmenšení prostoru k prostoru z minulého kroku. Ve sloupci **Sim.** je počet simulací určený uživatelem nebo danou metodou a ve sloupci **V tol.** je počet simulací, které program provedl. Rozdíl mezi počtem požadovaných a odsimulovaných simulací je počet simulací, které se se

Tabulka 4.9: Iterační postup hledání odhadu parametrů pro škrtící klapku

Ite.	**Met.**	**Sim.**	**V tol.**	**ko**	**k**	**b**	**J**	**MSE**	**Δ**
-	-	-	-	[Nm]	[Nm/rad]	[Ns/rad]	[gmˆ2]	**-**	[%]
1.	MC	100	100	0-0.4	0-2	0-0.6	0.01-15	**95.21**	-
2.	MC	100	100	0.04-0.3	0.6-2	0.1-0.55	0.01-15	**83.71**	-66
3.	MC	200	200	0.09-0.225	0.6-2	0.19-0.55	0.01-15	**58.45**	-58
4.	MC	400	388	0.1-0.225	0.6-2	0.26-0.45	0.01-15	**40.02**	-51
5.	MC	400	394	0.135-0.218	0.76-1.6	0.26-0.45	0.01-15	**34.08**	-60
6.	MC	400	363	0.17-0.21	0.858-1.25	0.26-0.35	1.6-12	**29.32**	-93
7.	MC	400	390	0.178-0.21	0.91-1.18	0.28-0.327	4.4-10.4	**24.91**	-83
8.	MC	1000	938	0.187-0.205	0.93-1.1	0.29-0.32	5.4-10	**24.55**	-83
9.	MC	1000	939	0.188-0.205	0.94-1.07	0.295-0.315	6.4-8.27	**24.45**	-80
10.	L-M	190	190	0.188-0.205	0.94-1.07	0.295-0.315	6.4-8.27	**24.34**	0
11.	MC	400	398	0.192-0.195	1.01-1.04	0.298-0.301	6.9-7.8	**24.34**	-99
12.	Grid	4096	4093	0.192-0.195	1.01-1.04	0.298-0.301	6.9-7.8	**24.33**	0
13.	L-M	527	527	0.192-0.195	1.01-1.04	0.298-0.301	6.9-7.8	**24.33**	0
14.	MC	980	980	0.192-0.195	1.01-1.04	0.298-0.301	6.9-7.8	**24.33**	0

svými parametry trefily do blízké oblasti tolerance. Oblast tolerance je okolí bodu (kombinace parametrů) v prohledávacím prostoru. Tato oblast je hyperkrychle, kde délky jednotlivých stran jsou učeny procentuálně k velikosti intervalu v daném rozměru. Do iterace 12. byla tolerance nastavena na 10%, další iterace měly toleranci vypnutou.

Nalezené parametry

V tabulce 4.10 jsou nalezené odhady parametrů pro jednotlivé metody. Z tabulky vyplývá, že nejmenší chyby pro nalezené parametry bylo dosaženo iterační metodou. Průběh nalezené nejmenší chyby během prohledávání je v grafu 4.11.

Porovnání odezvy naměřených dat a simulací s parametry, získanými pomocí jednotlivých metod, jsou v grafech 5.21, 5.22, 5.23.

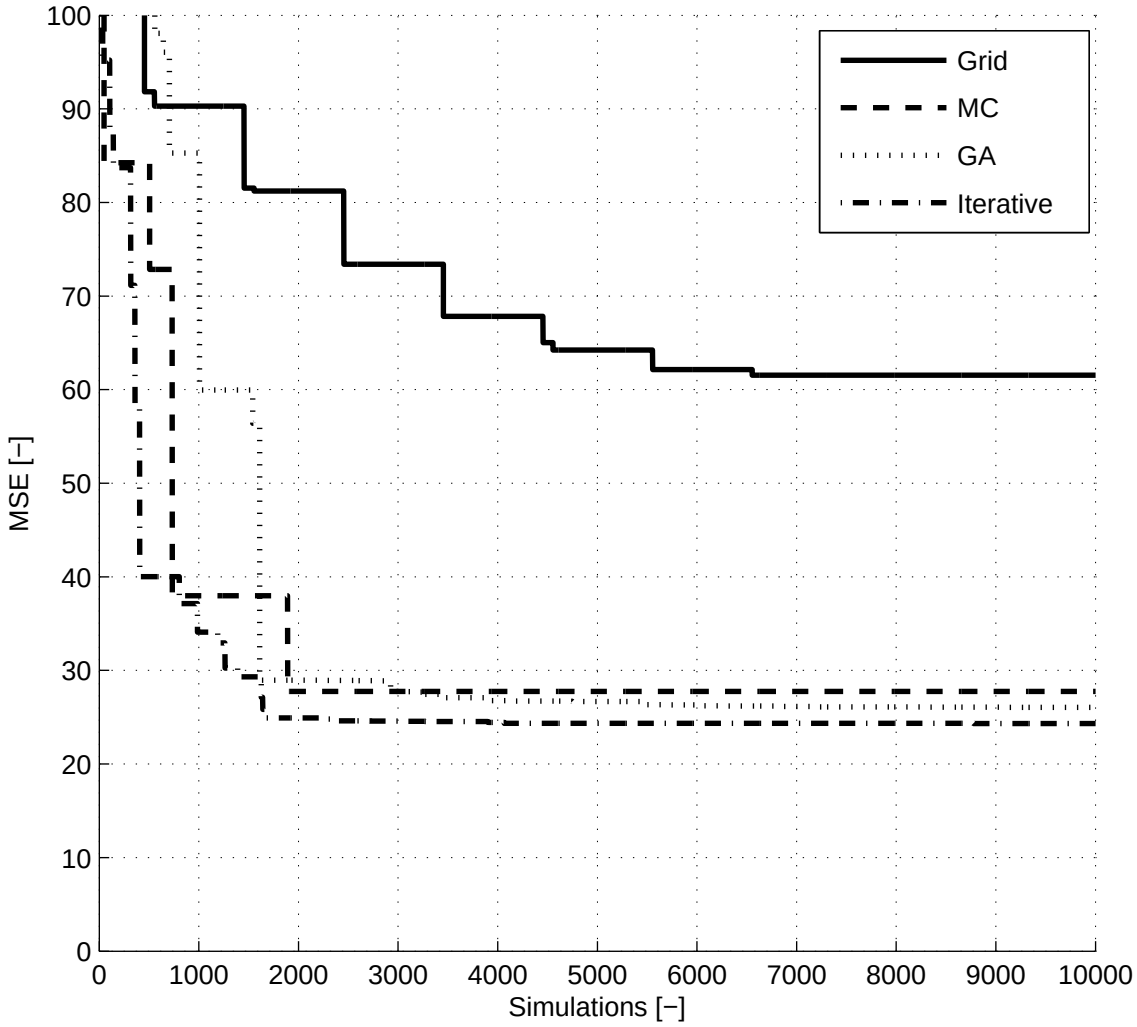
4.6.3 DC motor

V této ukázkové studii se bude porovnávat práce s programem PE a MPE. K tomuto porovnání poslouží DC motor, který je součástí výukové soustavy DoubleDrive. Na této soustavě byly naměřeny dva soubory dat na dvou různých experimentech. Soustava byla popsána rovnicemi 4.25 a 4.27.

Pro první experiment se naměřila odezva na motoru bez setrvačníku a bez zátěžového momentu $J_z = 0, M_z = 0$. Druhý experiment byl proveden bez zátěžového momentu a se se-

Tabulka 4.10: Tabulka odhadnutých parametrů pro škrtící klapku

Parametry	**J**	**b**	**k**	**k0**	**MSE**
Jednotky	[gmˆ2]	[Ns/rad]	[N/rad]	[Nm]	-
Hranice intervalu	<0.1; 15>	<0; 0.6>	<0; 2>	<0; 0.4>	-
Grid	10.003	0.333	1.111	0.178	61.534
Monte Carlo	6.095	0.283	1.059	0.189	27.745
Genetický Algoritmus	8.442	0.304	1.090	0.186	26.046
Iterační metoda	7.039	0.299	1.027	0.192	24.326



Obrázek 4.11: Minimální chyba v průběhu hledání

trvačníkem $J_z = 81.977\text{e}{-6} kgm^2, M_z = 0$.

Pro přidaný setrvačník je vypočítán moment setrvačnosti. Setrvačník se skládá z navijáku a dvou připevňovacích šroubů. Moment setrvačnosti navijáku byl získán z programu SolidWorks,

do kterého je pro výpočet mometu setrvačnosti kromě geometrie přidána také hustota materiálu. Pro šrouby a naviják se předpokládá homogenní hustota. Protože geometrie navijáku a šroubu je známá, stačí pro výpočet hustoty změřit hmotnost prvků . Objem navijáku je $O_n = 45241,67mm^3$, hmotnost $m_n = 363g$. Výsledná hustota $\rho_m = \frac{m_n}{O_n} = 8023.57kg/m^3$. Hmotnost šroubu byla pro větší přesnost získána pomocí měření deseti šroubů. Byla změřena na $m_s = 2.06g$. Objem šroubu je $O_s = 332mm^3$. Hustota šroubu je $\rho_s = \frac{m_s}{O_s} = 6204.82kg/m^3$. Celkový moment setrvačnosti přidaných těles k ose rotace je $J_z = 81.977\text{e}{-6}kgm^2$. Přidaný setrvačník změní dynamiku soustavy, ale neovlivní ostatní parametry.

Momentová rovnice DC motoru:

$$(J + J_z)\frac{d\omega}{dt} = M - b\omega - Tsgn(\omega) - M_z \tag{4.25}$$

Kde: $J[kgm^2]$ moment setrvačnosti rotoru, $J_z[kgm^2]$ moment setrvačnosti setrvačníku, $M[Nm]$ moment motoru, $M_z[Nm]$ zátěžový moment, $b[Nms/rad]$ koeficient viskózního tření, $T[Nm]$ koeficient suchého tření, $\omega[rad/s]$ úhlová rychlost rotoru.

Vztah mezi momentem a proudem kotvy:

$$M = K_m i \tag{4.26}$$

Kde: $K_m[Nm/A]$ momentová konstanta motoru, $i[A]$ proud motorem.

Napěťová rovnice DC motoru

$$L\frac{di}{dt} = U - e - Ri \tag{4.27}$$

Kde: $L[H]$ je indukčnost vinutí kotvy, $U[V]$ vstupní napětí, $e[V]$ zpětné elektromotorické napětí, $R[\Omega]$ odpor motoru.

Závislost zpětného elektromagnetického napětí na úhlové rychlosti rotoru.

$$e = K_u\omega \tag{4.28}$$

Kde: $K_u[Vs]$ napěťová konstanta motoru.

Vztah pro odpor kotvy:

$$R_a = R - R_i \tag{4.29}$$

Kde: $R_a[\Omega]$ je odpor vinutí kotvy, $R_i[\Omega]$ odpor zdroje.

Protože je elektromagnetická časová konstanta $\tau_a = \frac{L_a}{R_a}$ řádově menší, než elektromechanická časová konstanta $\tau_m = \frac{JR_a}{K_m K_u}$, lze člen $L\frac{di}{dt}$ v rovnici 4.27 zanedbat[13]. Protože je zátěžový moment nulový $M_z = 0$, je možné rovnice 4.27 a 4.25 upravit na:

$$(J + J_z)\ddot{\varphi} = U\frac{K_m}{R} - \left(\frac{K_m K_u}{R} + b\right)\dot{\varphi} - Tsgn\dot{\varphi} \tag{4.30}$$

Z rovnice vyplývá, že všechny parametry nejde jednoznačně určit. Pro hledání odhadu parametru se použijí náhradní parametry. V další úpravě se postup v programech PE a MPE rozchází.

Řešení úlohy pomocí PE

Protože PE neumí pracovat s více modely, je potřeba provést odhad parametrů dvakrát. Rovnice se upraví na dva vztahy, bez a s přidaným momentem setrvačnosti J_z.

Vztah bez přidaného momentu setrvačnosti:

$$\ddot{\varphi} = U\frac{K_m}{RJ} - \frac{1}{J}\left(\frac{K_m K_u}{R} + b\right)\dot{\varphi} - \frac{T}{J}sgn\dot{\varphi} \tag{4.31}$$

Vztah 4.31 se upraví na:

$$\ddot{\varphi} = p_1 U - p_2\dot{\varphi} - p_3 sgn\dot{\varphi} \tag{4.32}$$

Kde parametry p_1, p_2, p_3 jsou:

$$p_1 = \frac{K_m}{RJ} \tag{4.33}$$

$$p_2 = \frac{1}{J}\left(\frac{K_m K_u}{R} + b\right) \tag{4.34}$$

$$p_3 = \frac{T}{J} \tag{4.35}$$

Vztah s přidaným momentem setrvačnosti:

$$\ddot{\varphi} = U\frac{K_m}{R(J+J_z)} - \frac{1}{J+J_z}\left(\frac{K_m K_u}{R} + b\right)\dot{\varphi} - \frac{T}{J+J_z}sgn\dot{\varphi} \tag{4.36}$$

Vztah 4.36 se upraví na:

$$\ddot{\varphi} = p\acute{}_1 U - p\acute{}_2\dot{\varphi} - p\acute{}_3 sgn\dot{\varphi} \tag{4.37}$$

Kde parametry $p\acute{}_1, p\acute{}_2, p\acute{}_3$ jsou:

$$p\acute{}_1 = \frac{K_m}{R(J+J_z)} \tag{4.38}$$

$$p\acute{}_2 = \frac{1}{J+J_z}\left(\frac{K_m K_u}{R} + b\right) \tag{4.39}$$

$$p\acute{}_3 = \frac{T}{J+J_z} \tag{4.40}$$

Hledání odhadu parametrů p_1, p_2, p_3 se provede na datech z prvního měření bez přidaného momentu setrvačnosti. Odhad parametrů $p\acute{}_1, p\acute{}_2, p\acute{}_3$ se provede na datech z druhém měření s přidaným momentem setrvačnosti. K hledání odhadu parametrů se využije program PE. V obou případech se využije stejný model.

Program PE odhadl parametry $p_1 = 2942, p_2 = 11.592, p_3 = 310.85$ pro první experiment s chybou $SSE = 1120.9$. Pro druhý experiment odhadl parametry na $p\acute{}_1 = 1361.5, p\acute{}_2 = 22.238, p\acute{}_3 = 48.919$ s chybou $SSE = 13.941$.

Z nalezených parametrů lze zjistit hodnotu momentu setrvačnosti pomocí vztahu:

$$J = J_z\frac{p\acute{}_3}{p_1 - p\acute{}_3} = 81.977\text{e}{-6}\frac{48.919}{310.85 - 48.919} = 15.31\text{e}{-6} kgm^2 \tag{4.41}$$

Dále parametr T se vypočítá:

$$T = p_3 J = 310.85 \cdot 15.31\text{e}{-6} = 4.8\text{e}{-3} Nm \tag{4.42}$$

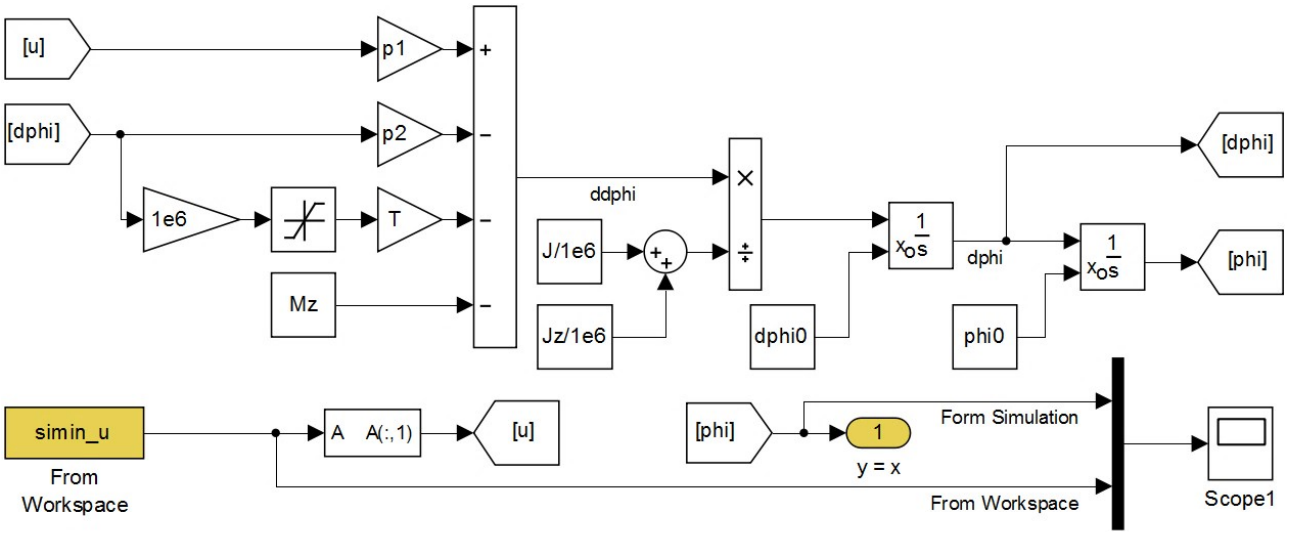
Parametry k_m a R nelze určit, ale je možné určit poměr $\frac{K_m}{R}$:

$$\frac{K_m}{R} = p_1 J = 2942 \cdot 15.31\text{e}{-6} = 0.0450 \tag{4.43}$$

Přibližně stejný výsledek by měl vyjít s použitím parametru $\acute{p_1}$:

$$\frac{K_m}{R} = \acute{p_1}(J + J_z) = 1361.5(15.31\text{e}{-6} + 81.977\text{e}{-6}) = 0.1325 \tag{4.44}$$

Přestože parametry K_m, R, J jsou pro oba experimenty totožné, výsledný poměr $\frac{K_m}{R}$ při použití parametrů $\acute{p_1}$ a p_1 se liší trojnásobně. Tedy nelze věřit odhadnutým parametrům. Důvodem je, že program PE pracuje vždy jen s jedním modelem, u kterého se snaží odhadnout parametry tak, aby chyba mezi měřením a simulací byla co nejmenší. Program tedy nedokáže najít kompromis, na rozdíl od programu MPE.



Obrázek 4.12: Model DC motoru pro odhad parametrů pomocí PE

Řešení úlohy pomocí MPE

Program MPE umí pracovat s více modely, tedy s více rovnicemi. Rovnice 4.30 se proto může upravit na dvě, 4.45 a 4.46, které sdílejí parametry p_1, p_2, T, J a konstantu J_z.

$$J\ddot{\varphi} = p_1 U - p_2 \dot{\varphi} - T sgn\dot{\varphi} \tag{4.45}$$

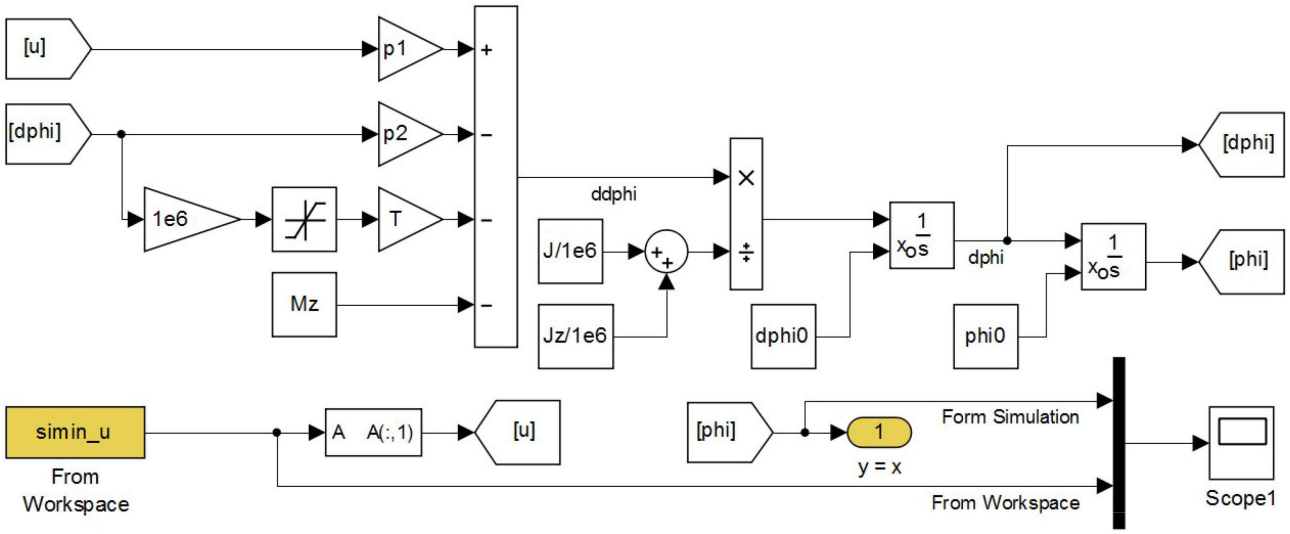
$$(J + J_z)\ddot{\varphi} = p_1 U - p_2 \dot{\varphi} - T sgn\dot{\varphi} \tag{4.46}$$

Kde parametry p_1, p_2 jsou:

$$p_1 = \frac{K_m}{R} \tag{4.47}$$

$$p_2 = \frac{K_m K_u}{R} + b \tag{4.48}$$

Program MPE odhadl parametry $J = 6.142\text{e}{-6} kgm^2, T = 4\text{e}{-3} Nm, p_1 = 2801.256, p_2 = 11.84$ s chybou $MSE = 26.17$. Stejně jako u řešení úlohy programem PE, nelze jednoznačně určit parametry K_m, K_u, R, b, pouze parametry J, T.



Obrázek 4.13: Model DC motoru pro odhad parametrů pomocí MPE

Porovnání výsledků

Protože chybu SSE z programu PE nelze porovnávat s chybou MSE z programu MPE, bylo potřeba získané parametry z programu PE ohodnotit v programu MPE. Společná chyba pro parametry získané z programu PE je $MSE = 118.46$. Chyba pro parametry získané z programu PE je $MSE = 26.17$.

5 Závěr

Začátek práce se věnuje programu PE v kapitole 2.1. Popis práce s programem, princip činnosti a uživatelské zkušenosti s programem jsou zhodnoceny v kapitole 2.5. Z této kapitoly vyplynuly nedostatky programu PE, jako jsou například: omezení počtu modelů, uvíznutí v lokálním minimu a mnoho dalších. Tyto nedostatky byly zohledněny v kapitole 3, kde se stanovily cíle pro nový program MPE.

V kapitole 4.1 je popsaná navržená struktura nového programu MPE. Struktura byla navrhnuta tak, aby zachovávala přehlednost a oddělila základní části programu do logických skupin.

Do programu byly implementovány prohledávací metody *Monte Carlo*, *Grid* a *Genetický algoritmus* a gradientní prohledávací metody *Levenberg-Marquardt*, *Trust region reflective* a *Trust region dogleg*. Tyto metody jsou součástí funkce *fsolve*. Program obsahuje i další prohledávací metody, které byly vytvořeny v jiné diplomové práci a rozšiřují tím schopnosti nového programu MPE.

Program využívá více metodik výpočtu chyby, které jsou popsané v kapitole 4.4.5. Program přináší možnost měnit metodiku určování chyby během procesu hledání, protože ukládá odezvu systému pro každou simulovanou kombinaci parametrů. Během procesu hledání lze taky měnit váhu pro jednotlivé kombinace modelů a souborů dat, a tím ovlivnit prohledávací algoritmus.

Součástí programu je vhodná vizualizace nalezených parametrů a odezvy podle stanovených cílů v kapitole 3.0.2. Jedná se hlavně o vizualizaci prohledávaného prostoru na intervalech jednotlivých parametrů. Tyto intervaly lze intuitivně zmenšovat, a tím zobrazit určitou oblast prohledávaného prostoru. Nalezené chyby lze taky různě seřadit pro vhodnou vizualizaci.

Pro program MPE byl vytvořen podrobný návod obsažený v příloze A. V manuálu je obsažen i návod na vytvoření nové prohledávací metody. Nově vytvořená prohledávací metoda lze přidat do programu MPE bez editace programu.

Důležitou součástí práce jsou příkladové studie, ve kterých se provádí hledání odhadu parametrů pomocí nového programu MPE. Příkladové studie ukazují schopnosti a výhody nového programu.

V první příkladové studii se hledají parametry nasimulovaného oscilátoru pomocí dvou experimentů, které sdílejí parametry. Pro nalezení odhadu parametrů byly využity metody vytvořené v této práci. V tabulce 4.8 jsou porovnány chyby pro odhadnuté parametry a v grafu 4.8 průběh nejmenší nalezené chyby během procesu hledání.

V druhé příkladové studii je provedeno hledání odhadu parametrů na reálné soustavě škrtící klapky. V této příkladové studii je ukázána síla iteračního zmenšování prostoru a změny prohledávací metody. Hodnoty nejmenší chyby pro jednotlivé metody, včetně iteračního přístupu, jsou v tabulce 4.10. Průběh nejmenší nalezené chyby během procesu hledání je v grafu 4.11.

V poslední příkladové studii je také provedeno hledání odhadu parametrů na reálné soustavě DC motoru. Jsou provedeny dva experimenty, které sdílejí parametry. V této příkladové studii je porovnán přístup pomocí programu PE a MPE, ze kterého vyplývá, že absence možnosti provádět hledání parametrů na více modelech vede k nalezení parametrů, které mají horší společnou chybu, než při použití programu MPE.

5.1 Náměty na další vývoj programu

Během práce na programu jsem přišel na další vlastnosti, které by mohl program MPE poskytnout. Některé byly implementovány do programu, ostatní jsou níže popsány a mohou posloužit pro další vývoj programu MPE.

Jak je uvedeno v kapitole 2.2.3, program PE má možnost zadat další vazby $C(x)$, pokud se s programem pracuje pomocí příkazů. Tato možnost v programu MPE chybí. Taková možnost by dovolila uživateli zohlednit jeho znalosti o soustavě, například vzájemné vazby jednotlivých parametrů.

Pro gradientní metody je důležitá volba kroku λ. V programu MPE ji nelze uživatelsky přívětivě měnit. Přidaná možnost editace této hodnoty by pomohla gradientním metodám.

Pokud pomocí iterační metody najde uživatel daný parametr s dostatečnou přesností, bylo by vhodné, aby měl možnost tento parametr změnit na konstantu. Změnou parametru na konstantu by se zmenšil počet simulací a také by se zrychlilo vykreslování nalezených parametrů. Změnu na konstantu lze použít, pokud se s programem MPE pracuje pomocí příkazů. Taková možnost v uživatelském rozhraní chybí.

Program MPE má stanovený počet míst, na která zaokrouhluje. To může vést ke ztrátě informací. Proto je nutné pro některé hledané parametry změnit model tak, aby se změnily jednotky daného parametru. Volba řádu zaokrouhlení by umožnila hledat parametry v základních jednotkách.

Program PE na rozdíl od programu MPE dovoluje upravit délku vstupního signálu, se kterým se provádí simulace. Pro program MPE je potřeba data před použitím oříznout. Vhodným rozšířením programu MPE by bylo přidání panelu *preprocessing*, který by se staral o ořezání a filtraci dat. V rámci takového panelu by byla i možnost upravit váhy na určité úseky signálu. Přidaná váha by umožnila využít metodiky výpočtu chyb, které s váhou pracují, jako například MSWD (Mean square weighted deviation).

Pokud by hodnoty parametrů byly zadávány včetně jednotek, mohl by program sám měnit rozsahy, to by vedlo k lepší představě o nalezené hodnotě. Například změnou kg na g. Takto určené hodnoty parametrů by byly pro uživatele představitelnější, také by se vyřešil problém se zaokrouhlením.

Literatura

[1] JAN, Jiří. *Číslicová filtrace, analýza a restaurace signálů*. Vyd. 1. Brno: Vysoké učení technické, 1997. ISBN 80-214-0816-2.

[2] MATLAB Help. *Mathworks* [online]. Natick, Massachusetts: The MathWorks, Inc., 2016 [cit. 2016-04-18]. Dostupné z: http://www.mathworks.com/help/matlab

[3] BARVÍŘ, Miroslav. *Modelování a identifikace: Určeno pro posl. fak. elektrotechn.* 1. vyd. Brno: VUT, 1991. Učební texty vysokých škol. ISBN 80-214-0302-0.

[4] NOSKIEVIČ, Petr. *Modelování a identifikace systémů*. 2000. Ostrava: Montanex, 1999. ISBN 80-722-5030-2.

[5] HUŠEK, Roman a Josef LAUBNER. *Simulační modely: celostátní vysokoškolská příručka pro studenty vysokých škol skupiny oborů 62 Ekonomické vědy*. 1. vyd. Praha: SNTL, 1987.

[6] BERGSTRA, James a Yoshua BENGIO. *Random search for hyper-parameter optimization*. Universite de Montreal, 2012.

[7] JANÍČEK, Přemysl. *Systémové pojetí vybraných oborů pro techniky: hledání souvislostí : učební texty*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2007. ISBN 978-80-7204-554-9.

[8] ZELINKA, Ivan. *Evoluční výpočetní techniky: principy a aplikace*. 1. vyd. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3.

[9] ZHOU WANG a A.C. BOVIK. Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. *IEEE Signal Processing Magazine* [online]. 2009, extbf26(1), 98-117 [cit. 2016-04-21]. DOI: 10.1109/MSP.2008.930649. ISSN 10535888. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4775883

[10] LEHMANN, E a George CASELLA. *Theory of point estimation*. 2nd ed. New York: Springer, c1998. Springer texts in statistics. ISBN 03-879-8502-6.

[11] VESELÝ, A. Návrh řídicího algoritmu pro elektromechanický aktuátor s výrazným suchým třením. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 54 s. Vedoucí diplomové práce doc. Ing. Robert Grepl, Ph.D.

[12] MALISZEWSKI, M. HIL simulace elektronické škrticí klapky. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 63 s. Vedoucí bakalářské práce Ing. Václav Sova.

[13] HANUŠ, P. Analýza a modelování sensorů a aktuátorů stavebnice LEGO Mindstorms. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 48 s. Vedoucí bakalářské práce doc. Ing. Robert Grepl, Ph.D.

[14] LJUNG, Lennart. Perspectives on system identification. *Annual Reviews in Control*. 2010, 34(1), 1-12. DOI: 10.1016/j.arcontrol.2009.12.001. ISSN 13675788. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/S1367578810000027

Seznam zkratek a symbolů

FSI	Fakulta strojního inženýrství
OOP	Objektově orientovaného programování
MPE	Mechlab's parametr estimation
PE, SPE	Parametr estimation
GUI	Graphical User Interface
MSE	Mean squared error
MAE	Mean absolute error
RMSE	Root mean squared error
NRMSE	Norm root mean squared error
PSNR	Peak signal to noise ratio
MSWD	Mean square weighted deviation
SSE	The sum of squares due to error
SAE	Sum absolute error
MPE_D	Mechlab's parametr estimation - data administration
MPE_M	Mechlab's parametr estimation - model administration
MPE_P	Mechlab's parametr estimation - definition of parametrs
MPE_S	Mechlab's parametr estimation - saving searched prametres and their simulations
MPE_GUI	Mechlab's parametr estimation - Graphical User Interface

Přílohy

A Uživatelský manuál k programu

S nástrojem MPE lze pracovat dvojím způsobem. Pomocí uživatelského rozhraní a nebo pomocí příkazů, kde je potřeba před spuštěním prohledávaní definovat cesty k modelům a datům a také nadefinovat meze pro hledání daného parametru. Stručný návod lze spustit příkazem: `help MPE_GUI`, kam se lze dostat přes odkazy ke všem funkcím.

A.1 Práce s MPE pomocí skriptu

Pro spuštění estimace pomocí skriptu je potřeba vytvořit a uložit skript ve složce, kde se nachází funkce MPE_GUI, MPE_D, MPE_M, MPE_P, MPE_S. Pro verzi Matlabu R2012b a starší také funkci `strsplit` nebo do jiné složky a přidat cestu k funkcím pomocí `addpath`, například: `addpath(strcat(pwd,'/Funs'));`

Administrace a úprava dat MPE_D

O administraci a úpravu dat se stará objekt MPE_D.

```
hD = MPE_D
```

Vytvoří objekt pomocí konstruktoru a vrací odkaz na tento objekt.

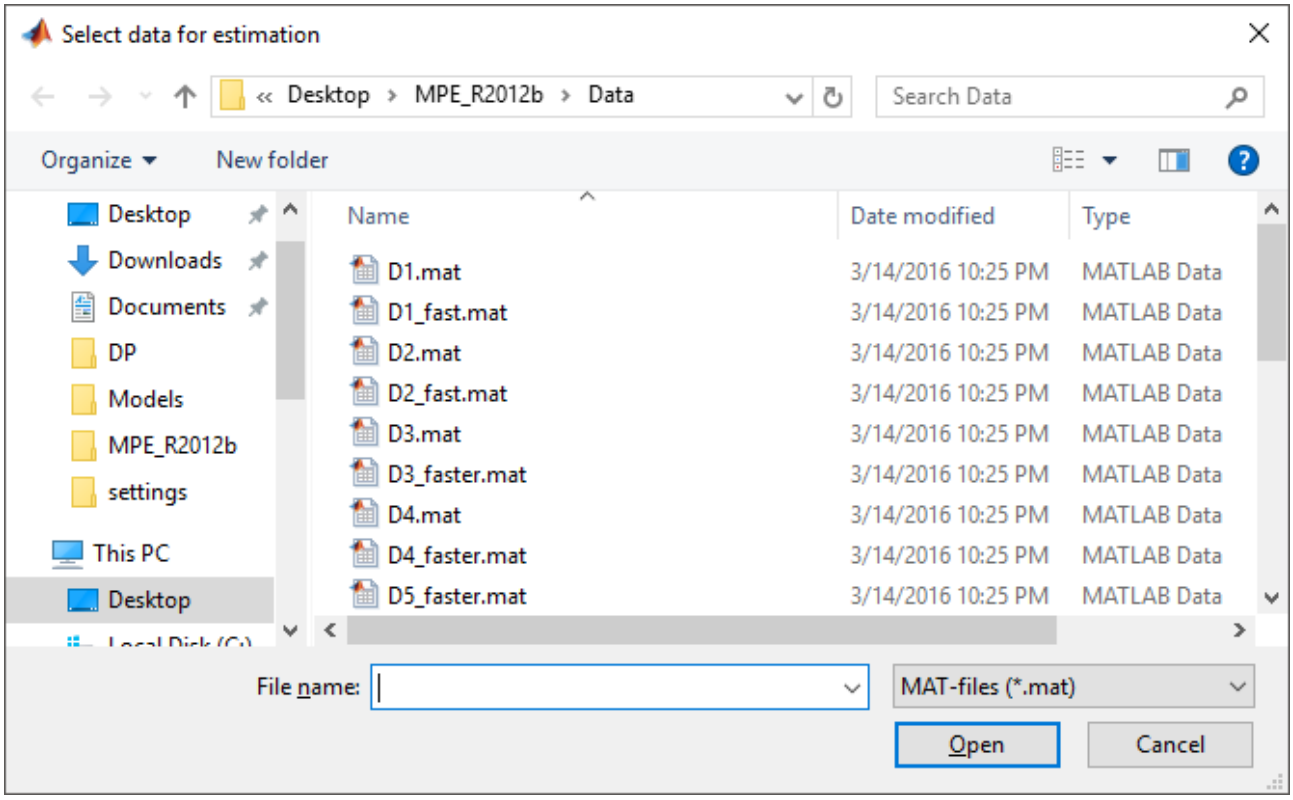
Přidání dat set_D

```
hD = set_D(MPE_D)
```

Spustí průzkumník pro určení dat k načtení. Viz obrázek 5.1 Nemusí se předtím vytvářet objekt pomocí konstruktoru.

```
hD.set_D
```

Spustí průzkumník pro určení dat k načtení. Viz obrázek 5.1 Lze vybrat jeden soubor dat a nebo více souborů dat najednou. Pokud objekt již datové struktury obsahuje, obsadí data nové pozice.



Obrázek 5.1: Vyběr dat

```
hD.set_D(path)
```

Načte data ze zadané cesty a uloží je na volnou pozici.

```
hD.set_D(path, decimate)
```

Načte data ze zadané cesty a uloží je na volnou pozici a provede decimaci. Hodnota zadané decimace určuje počet vzorků po decimaci.

```
hD.set_D(path, decimate, id)
```

Načte data ze zadané cesty, uloží je na zadanou pozici a provede decimaci. Decimace se neprovede, pokud se zadá nula místo decimace.

```
hD.set_D(name, y, u, time)
```

Vytvoří datovou strukturu ze vstupních vektorů. Vstupní vektor může mít libovolnou orientaci.

Administrace modelů MPE_M

O administraci modelů se stará objekt MPE_M.

```
hM = MPE_M
```

Vytvoří objekt pomocí konstruktoru a vrací odkaz na tento objekt.

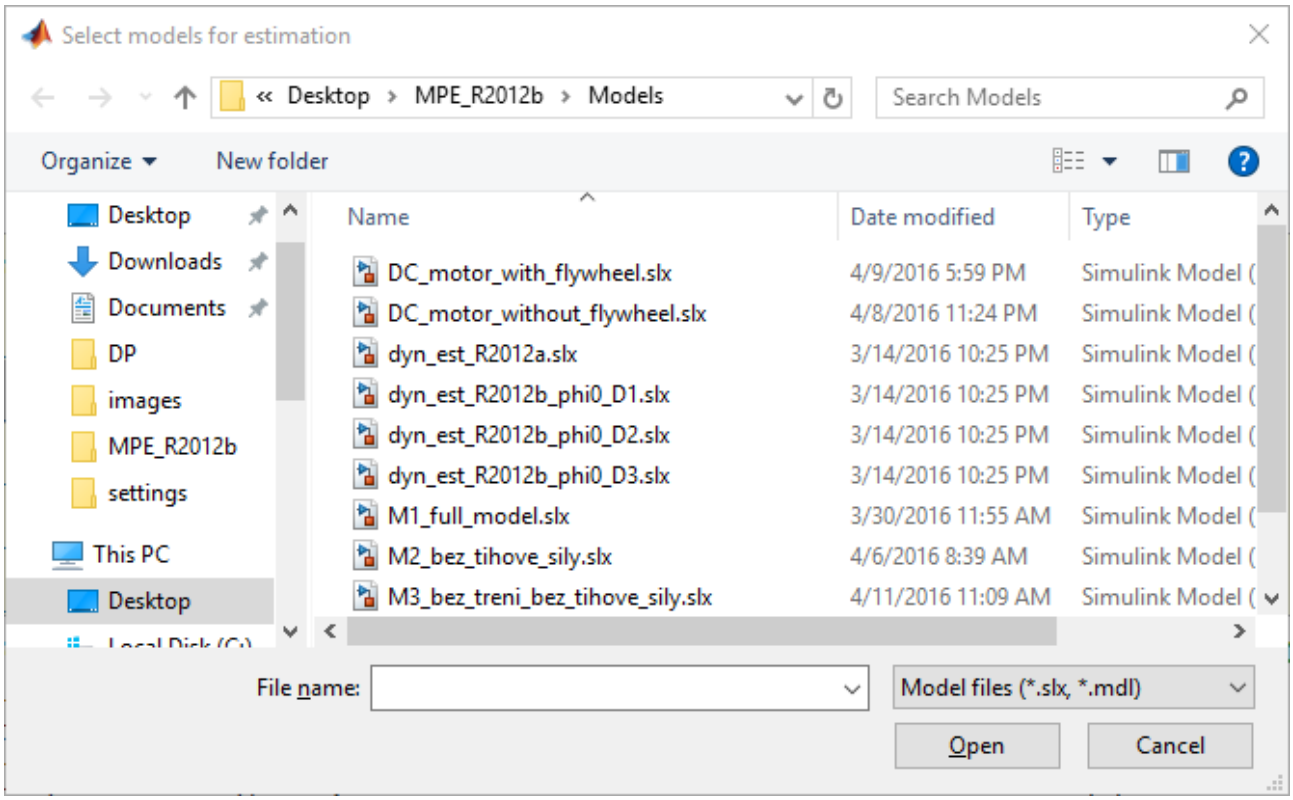
Přidání modelu set_M

```
hM.set_M(MPE_M)
```

Spustí průzkumník pro určení cesty k modelům. Viz obrázek 5.1 Nemusí se předtím vytvářet objekt pomocí konstruktoru.

```
hM.set_M
```

Spustí průzkumník pro určení cesty k modelům. Viz obrázek 5.1



Obrázek 5.2: Výběr modelu

```
hM.set_M(path)
```

Uloží cestu k modelu na volnou pozici.

```
hM.set_M(path, id)
```

Uloží cestu k modelu na zadanou pozici.

Nalezení parametrů v modelu set_parameters

```
hM.set_parameters
```

Funkce prohledá všechny modely, ke kterým je uložená cesta, vytáhne z nich parametry a uloží je. V případě, že je v modelu parametr uvedený vícekrát, je uložen jen jednou. Funkce pozná v modelu parametry, i když jsou součástí vzorce. Například $1/2*-dx0$. Parametry mohou obsahovat číselné znaky, jako třeba *dx0*. Lze je vypsat pomocí příkazu: `hM.table1.parameters`

```
hM.table{1}.parameters
'T'    'b'    'c'    'dx0'    'k'    'm'    'x0'
```

Přidání cesty pomocí tabulky set_path

```
hM.set_path({'M1',..,'Mn'},{'path1',..,'pathn'})
```

Uloží cesty k modelům z tabulky podle indexů, které jsou v tabulce uložené jako string ve tvaru M1,M2 atd. Tato funkce najde využití při ukládání dat z uitable v GUI.

Definice parametrů MPE_P

O definici parametrů se stará objekt MPE_P.

```
hP = MPE_P
```

Vytvoří objekt pomocí konstruktoru a vrací odkaz na tento objekt.

Přidání parametru set_P

```
hP.set_P(name, constant)
```

Do spodní i vrchní hranice intervalu uloží stejné číslo. Tímto způsobem se zadá konstanta. Pokud jméno konstanty již existuje, bude její hodnota přepsána.

```
hP.set_P(name, min, max)
```

Uloží spodní a vrchní hranice intervalu pro estimace. Pokud jméno parametru již existuje, budou hranice přepsány nově zadanými.

Přidání hranic intervalu z tabulky set_Min_Max

```
hP.set_Min_Max({'1:2',..,'1:2'})
```

Text z buněk tabulky rozdělí na část před dvojtečkou, kterou uloží do spodní hranice intervalu a část za dvojtečkou, kterou uloží do vrchní hranice intervalu. Funkce najde využití při ukládaní dat z tabulky v GUI.

Přidání názvů parametrů a konstant set_names

```
hP.set_names({'a',..,'z'})
```

Uloží názvy parametrů a konstant. Vhodné použití je s využitím funkce `hP.set_parameters`. Například: `hP.set_names(hM.table{1}.parameters)`.

Správa výsledků simulace MPE_S

O správu a úpravu výsledků simulací, kombinaci hledaných parametru a nalezené chyby pro jednotlivé kombinace parametrů se stará objekt MPE_P.

```
hS = MPE_P
```

Vytvoří objekt pomocí konstruktoru a vrací odkaz na tento objekt. Při tomto zápise zůstává metoda výpočtu chyby nastavena na výchozí metodě, a to na metodě MSE (Mean squared error).

```
hS = MPE_P(method)
```

Vytvoří objekt pomocí konstruktoru a vrací odkaz na tento objekt. Při tom se nastaví metoda, kterou se počítá chyba. Možnosti výpočtu chyby jsou: MAE, MSE, MPE, RMSE, NRMSE, PSNR. Více o metodě výpočtu chyby je v kapitole 4.4.5. Metodu lze změnit i později, a to příkazem: `hS.statistical_method = 'MAE'`.

Přidání kombinace dat a modelů pro estimaci set_S

```
hS.set_S(id_data, id_model)
```

Přidá kombinaci dat a modelu pro prohledávaní. Váha je při tomto zápisu nastavena na hodnotu 1.

```
hS.set_S(id_data, id_model, gain)
```

Přidá kombinaci dat a modelu s danou váhou pro prohledávaní parametrů. Váha slouží jako míra důvěry k dané kombinaci dat a modelu.

Výpočet společné chyby set_mse

```
hS.set_mse
```

Sečte chyby ze všech kombinací a podělí je počtem kombinací. Tato hodnota představuje společnou chybu pro všechny kombinace. Podle této hodnoty se prohledávací algoritmy rozhodují. Tato společná hodnota je průběžně ukládaná během prohledávaní. Příkaz je tedy potřeba použít jen při změně váhy pro určitou kombinaci dat a modelu nebo při změně metody výpočtu chyby.

Slučovaní výsledků prohledávání add_new_values

```
hS.add_new_values(tS, id_Data, id_Model)
```

Přidá ze struktury *tS* testované kombinace parametru *tS.p*, průběhy simulací *tS.simresult* a pro dané kombinace svoji i společnou chybu, přičemž je zkontrolováno, zda je délka jednotlivých částí struktury stejná. Tato funkce slouží pro spojování výsledků simulací tak, aby nedošlo k poškození dat.

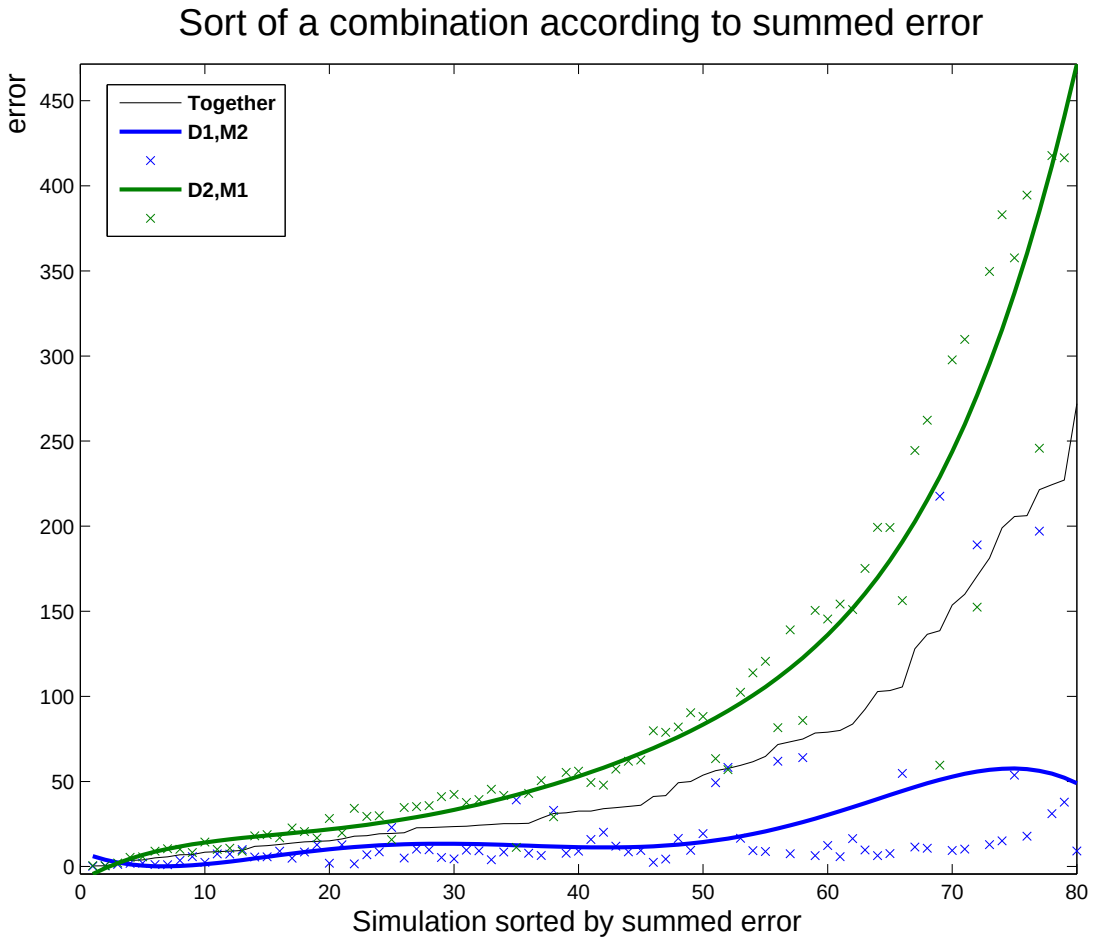
Vykreslení výsledné chyby seřazené podle společné chyby

comparison_by_summed_error

```
hS.comparison_by_summed_error
hS.comparison_by_summed_error(Level)
hS.comparison_by_summed_error(Level, percent, axes)
hS.comparison_by_summed_error(Level, percent, [], hS_1,..,hS_n)
hS.comparison_by_summed_error(Level, percent, axes, hS_1,..,hS_n)
```

Zobrazí graf, kde jsou chyby pro jednotlivé kombinace seřazené podle společné chyby. To znamená, že na vodorovné ose jsou jednotlivé kombinace seřazené podle hodnoty společné chyby. Jednotlivé body jsou proložené polynomem, kde jeho řád je určen hodnotou *Level*. Pokud není hodnota zadána, jedná se o polynom třetího řádu.

V grafu je vykresleno jen určité procento nejlepších hodnot tak, že při hodnotě *percent = 1* je zobrazeno 100% hodnot. Pokud není hodnota zadána, je zobrazeno 90% nejlepších hodnot. Zobrazením jen určitého množství nejlepších hodnot se zabrání znepřehlednění způsobenému extrémními hodnotami. Lze taky zadat osy *axes*, do kterých se graf vykreslí. Přidáním odkazů na další objekty MPE_S se provede porovnání těchto prohledávaní. Kliknutím levým tlačítkem myši lze změnit váhu dané kombinace dat a modelů. Tato nová váha je ihned zpětně přepočítána na všechny simulace. Viz obrázek 5.3

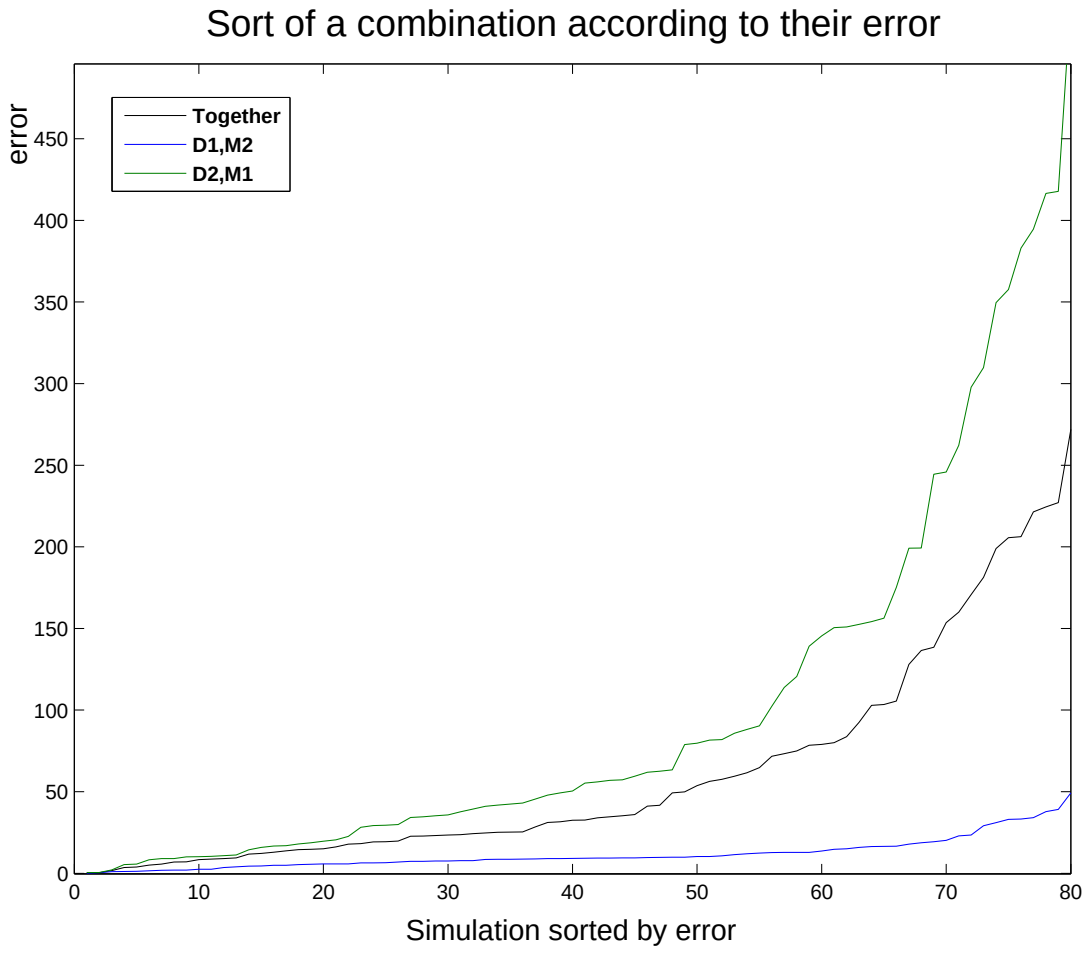


Obrázek 5.3: Vykreslení výsledné chyby: *hS.comparison_by_summed_error(5,0.8)*

Vykreslení seřazené podle výsledné chyby comparison_by_error

```
hS.comparison_by_error
hS.comparison_by_error(Level)
hS.comparison_by_error(Level, percent, axes)
hS.comparison_by_error(Level, percent, [], hS_1,..,hS_n)
hS.comparison_by_error(Level, percent, axes, hS_1,..,hS_n)
```

Zobrazí graf, kde jsou chyby pro jednotlivé kombinace seřazené podle svých chyb. To znamená, že na vodorovné ose jsou jednotlivé kombinace seřazené podle hodnoty chyby. Z grafu tedy nelze odečíst chybu pro určitou kombinaci parametrů. Jednotlivé body jsou proložené polynomem, kde jeho řád je určen hodnotou *Level*. Pokud není hodnota zadána, jedná se o polynom třetího řádu. V grafu je vykresleno jen určité procento nejlepších hodnot tak, že při *percent = 1* je zobrazeno 100% hodnot. Pokud není hodnota zadána, je zobrazeno 90% nejlepších hodnot. Zobrazením jen určitého množství nejlepších hodnot se zabrání znepřehlednění způsobenému extrémními hodnotami. Lze taky zadat osy *axes*, do kterých se graf vykreslí. Přidáním odkazů na další objekty MPE_S se provede porovnání těchto prohledávání. Viz obrázek 5.4

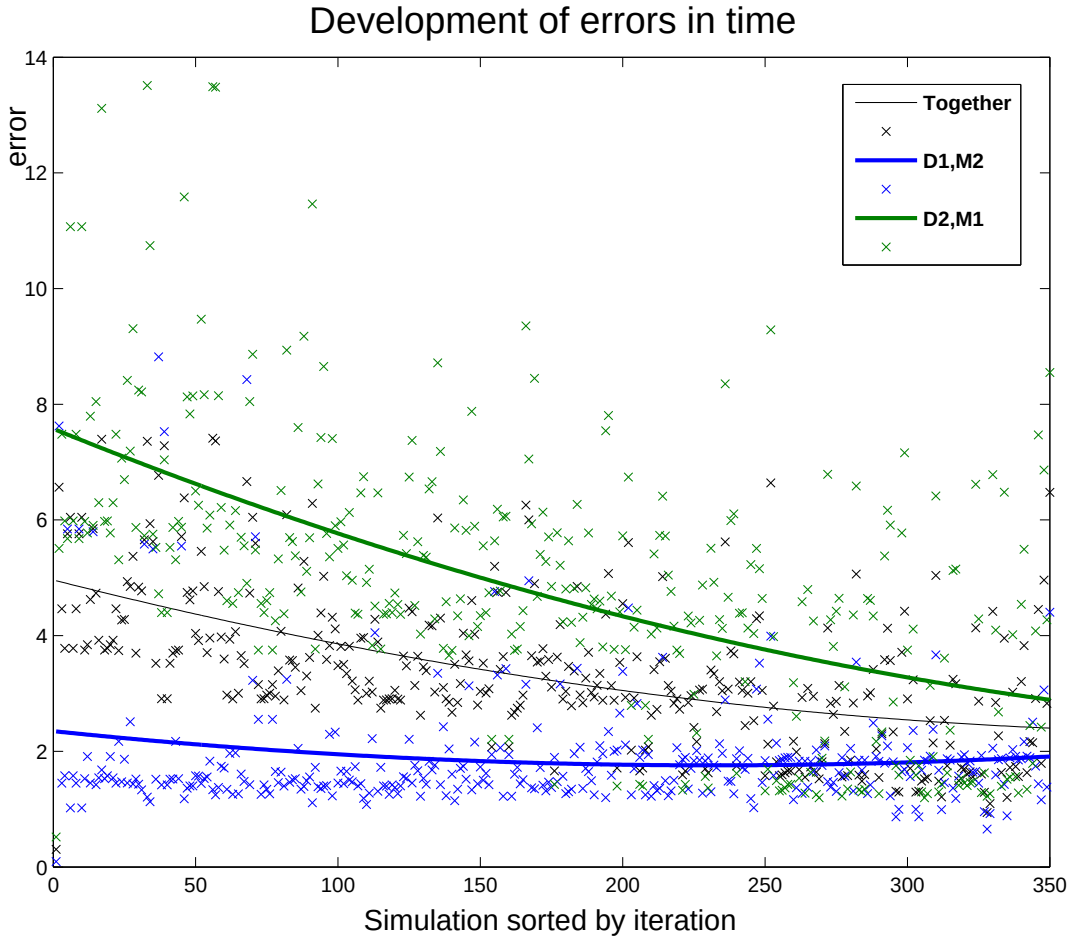


Obrázek 5.4: Vykreslení výsledné chyby: *hS.comparison_by_error(5,0.8)*

Vykreslení výsledné chyby podle pořadí simulací comparison_by_time

```
hS.comparison_by_time
hS.comparison_by_time(Level)
hS.comparison_by_time(Level, axes)
hS.comparison_by_time(Level, [], hS_1,..,hS_n)
hS.comparison_by_time(Level, axes, hS_1,..,hS_n)
```

Zobrazí graf, kde jsou chyby pro jednotlivé kombinace seřazené podle toho, jak byly simulovány. Jednotlivé body jsou proložené polynomem, kde jeho řád je určen hodnotou *Level*. Pokud není hodnota zadána, jedná se o polynom třetího řádu. Lze taky zadat osy *axes*, do kterých se graf vykreslí. Přidáním odkazů na další objekty MPE_S se provede porovnání těchto prohledávaní. Viz obrázek 5.5



Obrázek 5.5: Vykreslení výsledné chyby: *hS.comparison_by_time(2)*

Definice prohledávací metody MPE_GUI

O volbu prohledávací metody, úpravu nastavení této metody a následné zobrazení výsledků estimace se stará objekt MPE_GUI.

```
hMPE = MPE_GUI
hMPE = MPE_GUI(hD,hM,hP,hS,nameOfmetod);
hMPE = MPE_GUI(hD,hM,hP,hS,nameOfmetod,'PropertyName',PropertyValue,..);
hMPE = MPE_GUI(hD,hM,hP,hS,'MC','NumberOfSimulations',1000,...
                                'ShowSimulation','Off',...
                                'Tolerance',10);
```

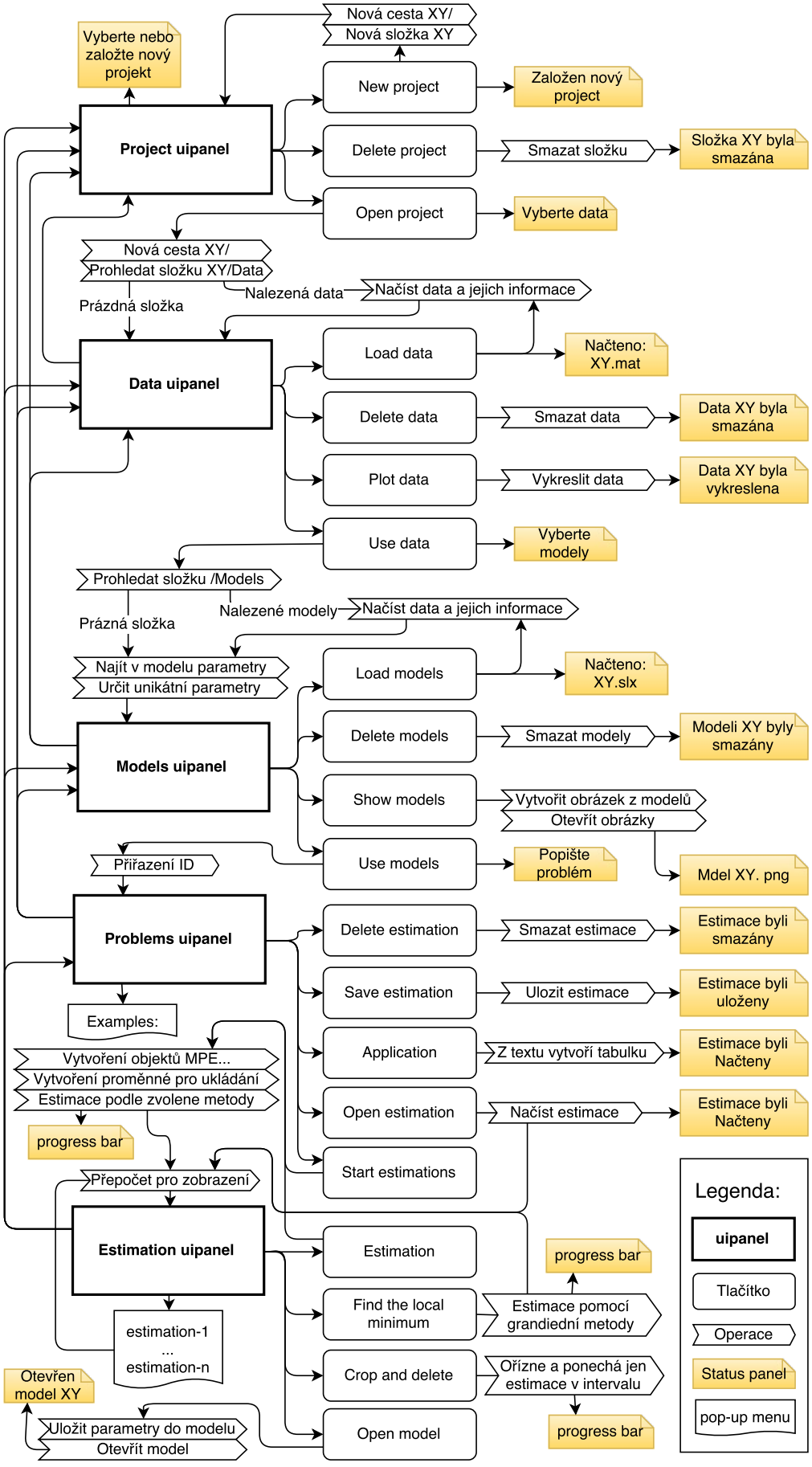
Pokud není zadán žádný parametr, spustí se GUI. Pokud je přidáno nastavení prohledávací metody, je toto nastavení upraveno. V jiném případě metoda pracuje s výchozím nastavením.

A.2 Práce s MPE pomocí uživatelského rozhraní

Program MPE má vlastní uživatelské rozhraní, které umožňuje intuitivně provést hledání parametrů. Před spuštěním programu MPE je potřeba připravit datovou strukturu do požadovaného tvaru a vytvoření modelu. Popis přípravy dat a modelů je popsán v kapitolách 4.3.1 a 4.3.2.

Uživatelské rozhraní je rozděleno do pěti částí. Projekt panel, Data panel, Model panel, Problém panel a Panel pro zobrazení výsledků. Uživatel je těmito panely veden postupně v pořadí, ve kterém byly vyjmenovány. Panely nelze přeskakovat, ale lze se vracet zpět k předchozím panelům pomocí uimenu. Strom programu, kterým je uživatel veden, je znázorněn na obrázku 5.6.

Ve spodní části okna programu se nachází informační panel, který během práce s programem informuje o krocích, které byly provedeny, nebo se od uživatele očekává, že je provede. Viz obrázek 5.7. Informační panel také graficky informuje o průběhu procesu hledání parametrů.



Obrázek 5.6: Strom uživatelského rozhraní

Spuštění programu MPE

Program se spustí příkazem `MPE_GUI`. Pro zdárné spuštění je nutné být ve složce */Funs*, kde se nachází objekty začínající MPE a složka */methods*. Pro větší přehlednost je vhodnější pracovat v kořenové složce programu a před spuštěním programu přidat cestu do složky */Funs*, a to třeba příkazem `addpath(strcat(pwd,'/Funs'));`.

Lze spustit více programů zároveň změnou odkazu `hMPE_GUI1=MPE_GUI; hMPE_GUI2=MPE_GUI;`. Takovéto spuštění více programů se nedoporučuje, protože pro simulaci využívá program *Workspace*. Tu mají potom programy společnou a mohlo by dojít k simulaci s nezamýšlenou hodnotou parametru.

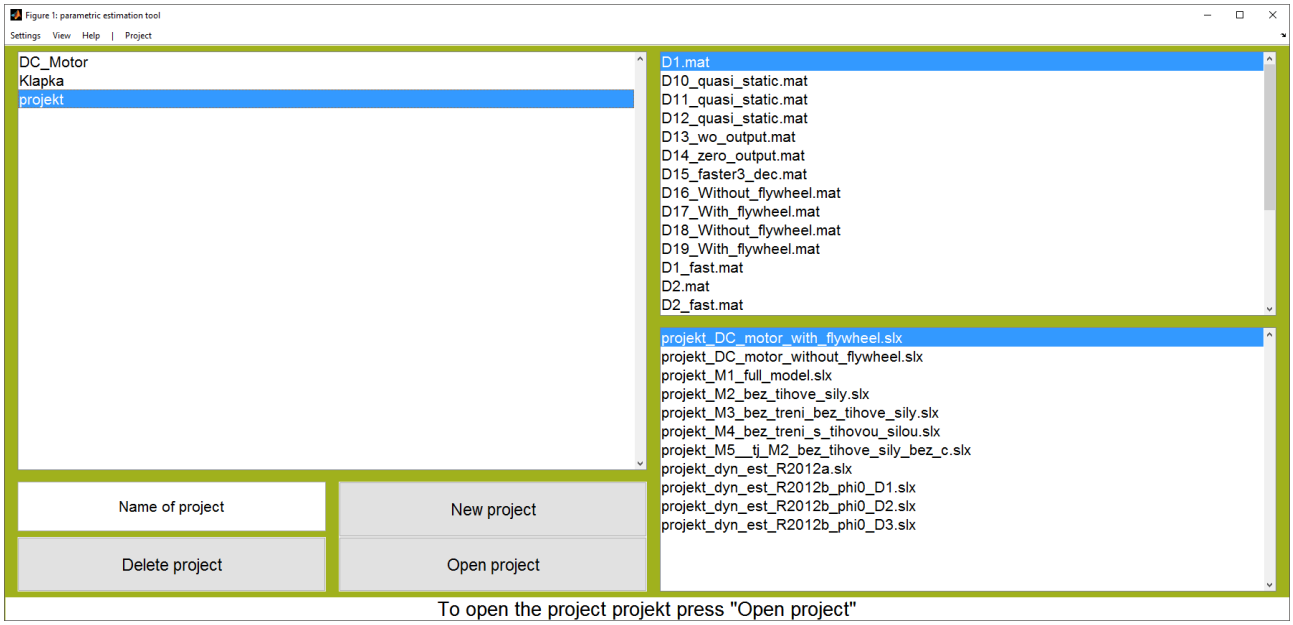
Po spuštění programu se prohledá složka */methods* a funkce v ní obsažené se uloží do objektu MPE_GUI z toho plyne, že přidaní nových prohledávacích metod je možné jen před spuštěním programu.

Vytvoření projektu

Prvním krokem po spuštění programu MPE bude vytvoření nebo volba projektu. Při vytvoření nového projektu se vytvoří nová složka s názvem projektu ve složce */Projects*. Název projektu by neměl obsahovat diakritiku a začínat číslicí. V této složce se také vytvoří podsložky */Data* a */Models*.

Projekty lze i smazat. Program při volbě smazat projekt po uživateli nevyžaduje potvrzení volby a smaže celou složku projektu. Projekt se nepřesouvá do koše systému.

Pro volbu projektu a pokračovaní do dalšího kroku je potřeba označit některý z projektů a zvolit volbu *Open projekt*. Pokud projekt již obsahuje data nebo modely, jsou zobrazeny v pravé části programu. Viz obrázek 5.7.



Obrázek 5.7: Volba projektu

Volba dat

Po volbě projektu je na řadě výběr dat pro hledání parametrů. Pokud v projektu nejsou nahrána žádná data, spustí se průzkumník pro vybrání nových dat. Viz obrázek 5.1. Tuto nabídku lze také vyvolat volbou *Load data*.

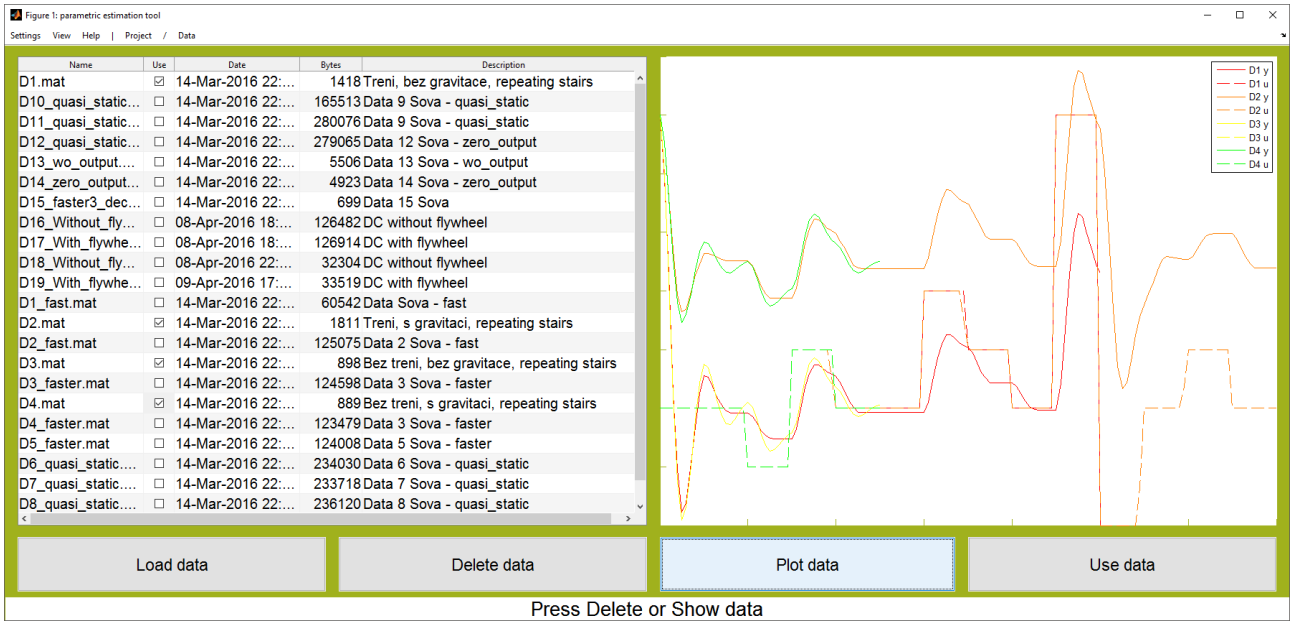
Data jsou překopírována z původního umístění do složky */Data*. Při změně dat se změna neprojeví na původních datech.

Kromě názvu, data poslední změny a velikosti je uveden i popis. Popis je uložen v datové struktuře do `D1{1,1}.name`. Popis jde editovat. Změna se uloží při klepnutí myší do jiné buňky.

Označením jednotlivých dat ve sloupci *use* se vyberou data, která se nabídnou pro hledání parametrů. Data výběrem dostanou svoje pořadí, které se počítá vzestupně od vrchní části tabulky. Nevybraná data se do číslovaní nezahrnou a v další části programu se neprojeví.

V tomto panelu lze vykreslit vybraná data volbou *Plot data*. Vstupní signál do modelu se vykreslí plnou čárou. Naměřená data se vykreslují přerušovanou čárou stejnou barvou jako vstupní signál viz obrázek 5.8. Vybraná data lze taky smazat volbou *Delete data*.

Po volbě dat se postoupí do další části volbou *Use data*. Zpět na volbu projektu lze pomocí kliknutí na *Projekt* v uimenu.



Obrázek 5.8: Volba dat

Volba modelů

Dalším krokem je výběr modelů. Pokud v projektu nejsou nahrány žádné modely, spustí se průzkumník pro výběr nových dat. Viz obrázek 5.1. Tuto nabídku lze také vyvolat volbou *Load data*. Modely jsou překopírovány do složky */Models*. Při změně modelů a jejich nastavení se změna neprojeví na původních modelech. Protože simulaci pomocí Simulinku lze spouštět jak pomocí cesty k modelu, tak i jenom názvem modelu, mohlo by dojít k chybě, pokud by více modelů mělo stejný název, i když by byly v různých složkách, do kterých má Matlab přidanou cestu. Z toho důvodů při překopírovaní dojde také k přejmenování tak, že se před název modelu přidá název projektu. Například: *Projekt_M1*. To zaručí jednoznačné určení simulovaného

modelu.

Program je vyvíjen a testován ve verzi MATLAB 2012b. V případě komplikací s kompatibilitou ve vyšší verzi Matlabu lze model uložit do předchozí verze. Při uložení z novější verze do verze MATLAB 2012b se odkomentují zakomentované bloky.

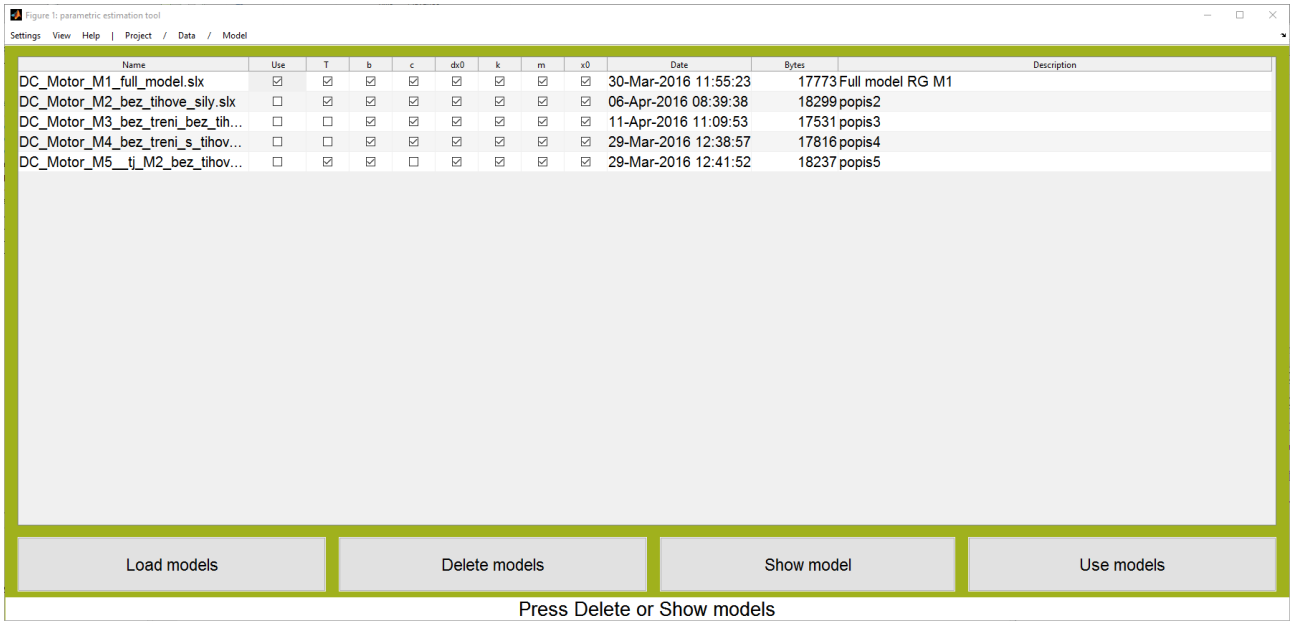
Modely musí obsahovat určité bloky, které jsou popsané v kapitole 4.3.2. Po případnou kontrolu lze modely rychle otevřít volbou *Show model*. Modely se otevřou ve výchozím prohlížeči obrázků operačního systému ve formátu *.jpg* nebo *.png* podle operačního systému. Takovéto zobrazení se provede mnohem rychleji než otevření modelů v Simulinku. Vytvořené obrázky modelů jsou uloženy ve složce */Models*, která se nachází ve složce projektu.

Při načtení se modely prozkoumají a vytáhnou se parametry v modelu obsažené. Parametry musí být jen v určitých blocích. Přesný popis přípravy modelů je popsán v kapitole 4.3.2. Všechny nalezené parametry jsou vypsané v tabulce ve sloupcích. V jednotlivých řádcích je znázorněno, které parametry jednotlivé modely obsahují.

Kromě názvu, data poslední změny a velikosti modelu je v tabulce také popis modelu, který lze editovat. Popis je uložen v samotném modelu.

Označením jednotlivých modelů ve sloupci *use* vyberete modely, které se nabídnou pro hledání parametrů. Modely výběrem dostanou svoje pořadí, které se počítá vzestupně od vrchní části tabulky. Nevybrané modely se do číslovaní nezahrnou a v další části programu se neprojeví.

Po volbě modelů se postoupí do další části volbou *Use models*. Zpět na volbu projektu lze pomocí kliknutím na *Projekt* nebo *Data* v uimenu.



Obrázek 5.9: Volba modelů

Definice problému

V tomto panelu se definuje samotné hledání parametrů. Tedy na kterých kombinacích dat a modelů se bude provádět hledání parametrů a které chyby kombinací se budou průměrovat. Také se určuje rozsah jednotlivých parametrů, nastavení hodnot pro konstanty, metoda prohledávaní prostoru a nastavení parametrů metody a hodnoty váhy.

V levé horní části programu se nacházejí dvě tabulky. Levá představuje vybraná data a

pravá vybrané modely. Důležitý je levý sloupec, který přiřazuje identifikační číslo dat anebo modelů ve formátu: písmeno D a číslo nebo písmeno M a číslo. Tímto identifikačním číslem se definuje problém. Slovem problém je myšleno, na kterých kombinacích dat a modelu se provede hledání parametrů a jakým způsobem se bude pracovat s výslednou chybou.

V pravé horní části programu se nachází textové pole, kde uživatel definuje problém. Zde symbol čárky *","* rozděluje identifikační číslo dat a modelů. Symbol plus *"+"* určuje, že výsledná chyba se v rámci sčítanců chyby kombinací průměruje. To ovlivňuje metodu prohledávání. Více v kapitole 4.4.3. Dalším symbolem je středník *";"*, který rozděluje jednotlivé hledání parametrů. Jednotlivá hledání spolu nijak nesouvisí a mohou používat různé metody nastavení, ale i parametry. Tímto způsobem lze porovnávat různé metody prohledávaní, protože program umožňuje porovnávat prohledávaní mezi sebou. Poslední možností je použít symbol hvězdičky *"*"*, který určuje váhu pro danou kombinaci. Tímto způsobem lze určit míru důvěry pro danou kombinaci. Míra váhy se píše před symbolem. Pro použití těchto symbolů je potřeba dát kombinace dat a modelů do závorek.

Pomůckou pro definici problému je pop-up menu, kde lze zvolit jeden z příkladů definice problému. Příklad se vypíše do textového pole, kde se problém definuje a kde lze upravit.

Volbou *Application* se vytvoří ve spodní části programu tabulky jednotlivých hledání, které byly definovány v textovém poli. Při změně textového pole a znovu volby *Application* se tabulky smažou a vytvoří se nové. Pro každou tabulku hledání bude vytvořen nový objekt.

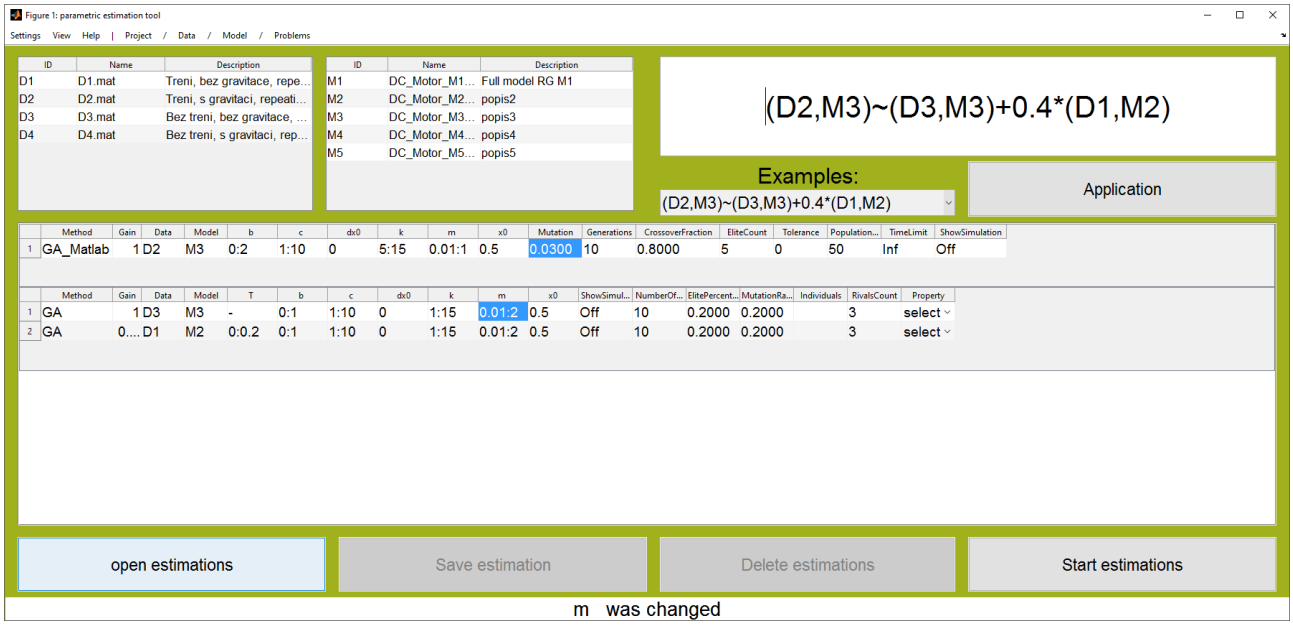
V tabulce lze vybrat prohledávací metodu, velikost váhy pro danou kombinaci, identifikační číslo dat a modelů, rozsah parametrů a nastavení pro danou metodu. Pro volbu nastavení metody je potřeba nejdříve vybrat metodu. Metodu lze vybrat jen jednou. Změnit metodu po jejím výběru lze pouze volbou *Application*. Výběrem metody se do rolovací nabídky posledního sloupce *Property* přidají názvy nastavení pro vybranou metodu. Každá metoda má rozdílné nastavení a z tohoto důvodu nelze metodu po vybrání měnit. Výběrem některého názvu nastavení z posledního sloupce se vytvoří nový sloupec s názvem vybraného nastavení. Jako hodnota se zobrazí výchozí hodnota nastavená v metodě. Tuto hodnotu lze nyní upravit. Nevybrané nastavení zůstane nastaveno na výchozí hodnoty pro danou metodu. Výběr nastavení lze opakovat, než se vybere poslední z nabídky nastavení.

Parametr definuje minimální a maximální hodnotou rozsahu rozdělení symbolem dvojtečky *":"*. Jednotlivé kombinace v rámci jednoho hledání musí mít stejný rozsah. Program nedovolí různý rozsah parametrů v rámci jednoho hledání parametrů. Hodnotu tedy stačí nastavit jen pro jednu kombinaci.

Konstanta se nastaví zadáním její hodnoty do příslušné buňky, jako u parametrů *dx0* a *x0* na obrázku 5.10. Pokud metoda používá pro svůj výpočet hodnotu počtu parametrů, konstanty se do tohoto výpočtu nezahrnují. Konstanty se taky nezobrazí ve vizualizaci v dalším panelu.

Lze také použít volby *open estimation* pro načtení dřívějšího hledaní. Po načtení se rovnou zobrazí vizualizace. Po načtení lze s pokračovat s prohledáváním a tedy pokračovat na dřívějším projektu. Také lze hledání parametrů smazat nebo uložit, pokud už nějaké hledání parametrů proběhlo.

Volbou *Start estimations* se vytvoří objekty MPE_D, MPE_M, MPE_P a MPE_S. Přepne se na nový panel a spustí se prohledávání. Proces prohledávání zobrazuje progress bar. Informace o postupu hledání se obnovuje po celém procentu z důvodů šetření výpočetního výkonu. Je zobrazován také čas do konce prohledávaní, který počítán pomocí doby, za kterou se provedly už proběhlé simulace. Ze začátku může být čas nepřesný, ale s nárůstem procent se čas zpřesňuje. Pokud je zapnutá vizualizace, tak se zobrazují právě simulované parametry.



Obrázek 5.10: Definice problému

Vizualizace

Po dokončení hledání parametrů se zobrazí v levé části rozsahy parametrů a v pravé části porovnání průběhu simulací k naměřenému průběhu. Zobrazí se tisíc nejlepších výsledků z důvodu rychlé odezvy programu. Lze zobrazit i více než tisíc simulací úpravou prvního intervalu, který zobrazuje společnou chybu kombinace parametrů pro všechny kombinace dat a modelů.

V levé části programu se nachází intervaly parametrů, kde jednotlivé modré čárky představují konkrétní hodnotu daného parametru v daném rozsahu. Čím je čára tmavější, tím výsledná chyba byla menší, tedy lépe odpovídal výstup ze simulace měřením.

V pravé části je výsledný průběh simulací těch kombinací parametrů, které jsou uvnitř intervalů na levé části programu. Čím je čára tmavější, tím výsledná chyba byla menší a tedy lépe odpovídal výstup ze simulace měřením. Zelená čára představuje měření.

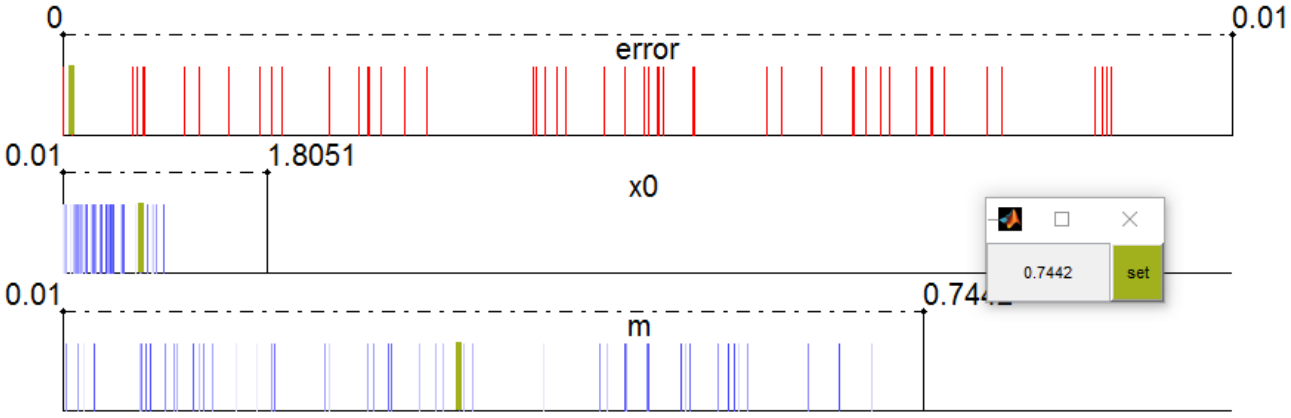
V pravém grafu je legenda, kde momentálně zobrazená kombinace dat a modelů je zvýrazněna červenou barvou. Na kombinace v legendě lze kliknout a tím přepnout zobrazenou kombinaci.

Pod pravým grafem je pop-up menu, ve kterém lze vybrat proběhlé hledání parametrů. Po změně výběru dojde k překreslení levého i pravého grafu.

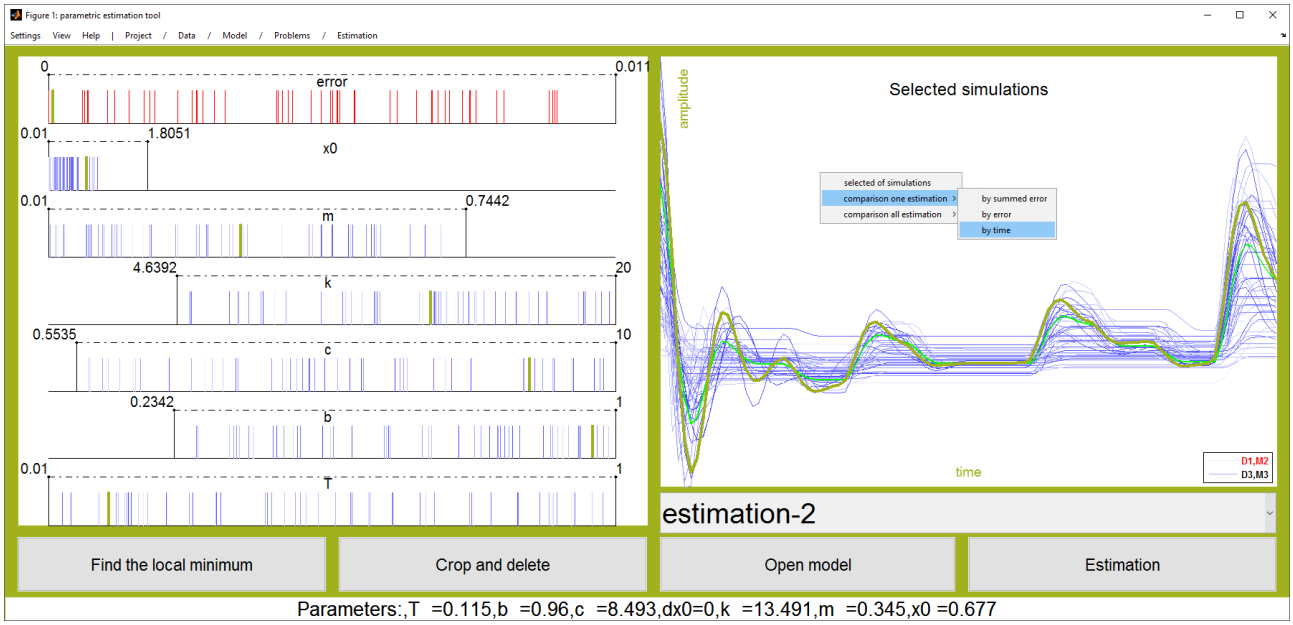
Po kliknutí do levého grafu se přesune příslušná hranice intervalu na tuto pozici. Posune se ta hranice, která se nachází nejblíže. Po kliknutí na levý nebo pravý okraj levého grafu se posune maximální nebo minimální hranice intervalu na výchozí hodnotu. Kliknutím na hodnotu se zobrazí okno (5.11), kde lze nastavit přesnou hodnotu hranice intervalu. Po volbě *set* se uloží nová hranice intervalu. Třetí možností, jak upravit interval, je kliknutí na některou z hranic intervalu, pohybem myši najet na zvolenou hodnotu a kliknutím potvrdit. Při pohybu myší se změní kurzor na obousměrnou vodorovnou šipku a v místě kurzoru se zobrazí červená čára, která představuje novou hranici intervalu. Pokud se s kurzorem vyjede mimo interval, operace se ukončí.

Po změně intervalu se obnoví vykreslené intervaly ve všech intervalech tak, aby byly zobrazené jen ty kombinace parametrů, které se nachází vevnitř všech intervalů. Jako na obrázku 5.12. Změna intervalu se taky projeví na případném novém hled8ní parametrů, které bude provedeno v tomto novém prostoru ohraničeném novými hranicemi.

Hodnoty parametru lze zobrazit kliknutím na libovolnou čáru, která představuje hodnotu



Obrázek 5.11: Nastavení nové hranice



Obrázek 5.12: Vizualizace

parametru v levé části programu nebo na průběh simulace v pravém grafu. Po výběru se tmavě zelenou barvou zvýrazní všechny hodnoty parametrů dané kombinace a hodnota chyby v prvním intervalu. Také se zvýrazní průběh dané simulace. Ve spodním informačním panelu se vypíšou hodnoty všech parametrů a konstant pro danou simulaci. Zvýraznění kombinace je zobrazeno na obrázku 5.12.

V pravém grafu lze vyvolat kontext menu, kde lze přepínat, co bude vykresleno v pravém grafu. Viz obrázek 5.11 Na výběr je *Selected of simulation* (zobrazení průběhů simulací), *comarison one estimation by summerd error* (Zobrazení chyby jednotlivých kombinací dat a modelů seřazené podle společné chyby), *comarison one estimation by error* (Zobrazení chyby jednotlivých kombinací dat a modelů seřazené podle chyb jednotlivých kombinací), *comarison one estimation by time* (Zobrazení chyby jednotlivých kombinací dat a modelů seřazené podle toho, v jakém pořadí probíhala simulace), *comarison all estimation by error*(Zobrazení chyby jednotlivých kombinací dat a modelů seřazené podle chyb jednotlivých kombinací v rámci všech procesů hledání parametrů) a *comarison all estimation by time*(Zobrazení chyby jednotlivých kombinací dat a modelů seřazené podle toho, v jakém pořadí probíhala simulace v rámci všech

procesů hledání parametrů). Tyto grafy lze vyvolat i pomocí příkazu. Syntaxe příkazu a podrobnější popis vizualizace těchto grafů je v kapitole A.1.

Volbou *Estimation* se provede nové prohledávaní parametrů, ale pouze toho prohledávaní (estimace), která je v té chvíli vybraná. Pro spuštění všech nadefinovaných hled8ní estimací je potřeba se vrátit do předchozího panelu *Problems* pomocí uimenu a volbou *Start estimations*. Nové hledání parametrů se provede jen v prostoru ohraničeném pomocí hranic intervalů. Zmenšovaním intervalů se tedy zmenšuje prostor, ve kterém se kombinace parametrů hledá. To vede k efektivnějšímu hledání. Iterační přístup hledání je popsán v kopitole 4.4.4.

Volba *Crop and delete* ořízne v objektu MPE_S výsledky simulací podle rozsahů zvoleních intervalů. Tento výřez zanechá a zbytek simulací vymaže. To bude mít pozitivní vliv na reakci programu při velkém počtu simulací. Uživatel není vyzván k potvrzení této volby a smaz8ní dat je trvalé a nelze je vzít zpět.

Volba *Find the local minium* spustí hledání parametrů pomocí vybrané gradientní metody prohledávání. Více o gradientních metodách použitých v programu MPE je v kapitole 4.5.4. Proces najde nejbližší lokální minimum. Počáteční hodnoty jsou kombinace parametrů, které jsou uvnitř všech intervalů. Jedno hledání může trvat i několik minut podle toho, jak je počáteční poloha vzdálena od lokálního minima. Z tohoto důvodu je vhodné vytvořit pomocí intervalů jen malé množství počitatelných poloh. Při hledání může být nalezeno minimum mimo interval, to je způsobeno tím, že gradientní metoda hranice prohledávacího prostoru nepotřebuje a proto jimi není zbytečně omezená.

Další volba je *Open model*, která momentálně vybraný model otevře. Před otevřením se do Model Properties/Callbacks/PreLoadFcn uloží hodnoty parametru kombinace s nejmenší chybou. Viz obrázek 5.13. Funkce *PreLoadFcn* se provede před načtením modelu v Simulinku. Ve funkci je kromě hodnot parametrů i cesta datům. Takto vytvořená funkce dovoluje spustit model v Simulinku, který provede simulaci s parametry, odpovídajími kombinaci parametrů, jenž dosáhla nejmenší chyby. V bloku *Scope* lze porovnat naměřená a odsimulovaná data. Protože je funkce uložená uvnitř modelu, lze model přesunout do jiné složky a spustit bez programu MPE. Tímto způsobem lze rychle aplikovat nalezené řešení do modelu a hned se zabývat regulací bez nutnosti přepisovat nalezené výsledky.

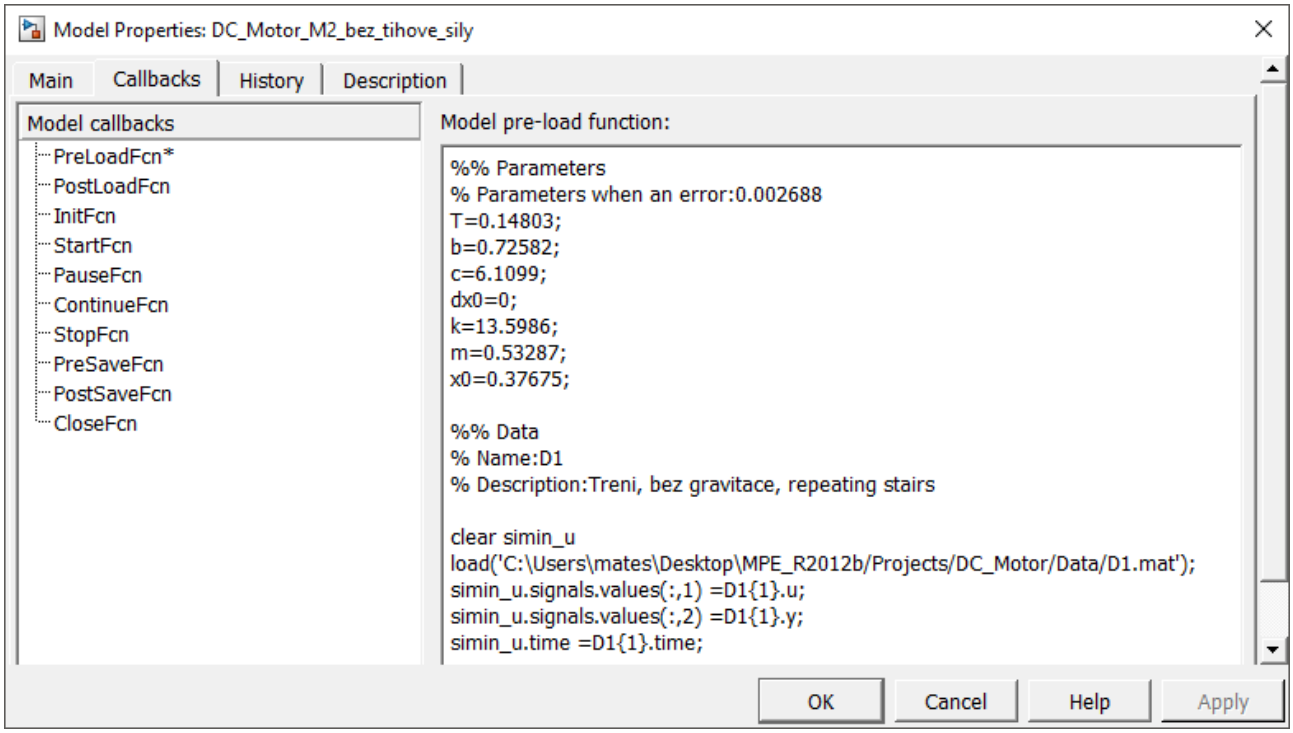
Možnost otevřít model a provést simulaci v prostředí Simulink je taky vhodné pro dobrou volbu řešiče. Volba řešiče má vliv na přesnost simulace a dobu simulace.

A.3 Přidání nové prohledávací metody do MPE

Program MPE je vytvořen tak, aby umožnil jednoduché vytvoření nových prohledávacích metod. Pokud je dodržena struktura funkce prohledávací metody, stačí danou metodu přesunout do složky */Funs/methods*. Program se postará o přidání metody do seznamu a také o zjištění parametrů (nastavení) metody. Prohledávací metoda se stará pouze o určování nových kombinací parametrů, které budou simulovány. O ukládaní vizualizací a ohodnocování simulace se stará program MPE.

Struktura prohledávací metody

Funkce musí obsahovat hlavní funkci (která je definovaná jako první), funkci `m_solve` a libovolný počet dalších funkcí. Hlavní funkce musí mít stejný název, jako název m-souboru. Tímto názvem se metoda definuje v programu MPE.



Obrázek 5.13: Parametry v PreLoadFcn

```
function funs = name_of_method
    funs.m_solve = @m_solve;
    funs.properties = { 'NumberOfSimulations',  100;...
                        'ShowSimulation',       'Off';...
                        'Tolerance',            0};
end
%% Solve
function m_solve(hObject,in_NumData,in_NumModel,properties)
    ...
end
%% function_1 ... function_n
function  [output] = function_1(input)
    ...
end
...
function  [output] = function_n(input)
    ...
end
```

Hlavní funkce musí obsahovat příkaz `funs.m_solve = @m_solve`, který přidává odkaz na funkci `m_solve` do struktury `funs`. Do této struktury se také přidává tabulka. Tabulka definuje v prvním sloupci název parametru (veličina, se kterou metoda operuje, nikoliv hledaný parametr)

a v druhém sloupci výchozí hodnota pro daný parametr. Ve druhém sloupci tabulky se nemusí nacházet jen číselná hodnota, ale i textový řetězec, vektor či matice. V tabulce musí zůstat ve druhém řádku parametr `'ShowSimulation'`. Tento parametr určuje, zda bude během simulace vizualizovat průběh hledání. Zapnutá vizualizace výrazně zpomaluje hledání parametrů, proto je vhodné vizualizovat hledání parametrů jen při tvorbě nové prohledávací metody.

Funkce `m_solve` je volána programem MPE. Prohledávání neskončí, dokud tato funkce není provedena. Funkce má striktně danou syntaxi:

```
function m_solve(hObject,in_NumData,in_NumModel,properties)
    ...
end
```

Kde *hObject* je odkaz na objekt programu MPE. Proměnná *in_NumData* je vektor, ve kterém jsou identifikační čísla dat pro hledání. Stejně jako proměnná *in_NumModel*, která obsahuje identifikační čísla modelů. V proměnné *properties* je uživatelem upravená tabulka parametrů, definovaná v hlavní funkci.

Ohodnocovací funkce

Ohodnocovací neboli fitness funkce je funkce, která pomocí simulace určuje, jak je navržená kombinace parametrů úspěšná. Tedy vrací hodnotu, která představuje míru shody naměřených dat a simulace při navržených hodnotách parametrů. Čím je číslo menší, tím je větší shoda měření a simulace. Principem prohledávací metody je minimalizovat výstupní hodnotu z vyhodnocovací funkce vhodnou volbou navržených parametrů.

```
[~,error]=hObject.fitness_function(in_NumData,in_NumModel,properties,tS);
%% Nebo
hObject.fitness_function(in_NumData,in_NumModel,properties,tS);
error=hObject.hS.error(end);
```

Proměnná *in_NumData* je vektor, ve kterém jsou identifikační čísla dat pro hledaní. Stejně jako proměnná *in_NumModel*, která obsahuje identifikační čísla modelů. V proměnné *properties* je tabulka nastavení. Strukturou *ts.p* definujeme kombinaci navržených parametrů, kde řádky představují parametry. Lze simulovat buď jednu kombinaci nebo skupinu kombinací. Výstupní parametr *error* určuje společnou chybu všech kombinací.

Podobnost hledaných parametrů (is_in_p)

Doba simulace jedné kombinace je obrovská ve srovnání s dobou, kterou zabere výpočet nutný pro navržení nové kombinace pro simulaci. Z tohoto důvodu je vhodné omezit provádění zbytečných simulací. Pokud daná kombinace s určitou přesností už byla odsimulovaná, nová simulace nepřinese nové informace o prohledávaném prostoru.

```
is_in_p=hS.is_in_p(new_combinations,Tolerance,hObject.hP);
```

Parametr *new_combinations* obsahuje kombinace, které algoritmus navrhl pro simulaci. Míra přesnosti shody kombinací se zadává proměnou *Tolerance* v procentech. Parametr *hObject.hP* určuje množinu kombinací, ve které se hledá shoda. Výstupem z funkce `is_in_p` je vektor nul a jedniček stejně dlouhý, jako počet zkoumaných kombinací. Jednička znamená, že daná kombinace parametrů se nachází ve zkoumané množině kombinací. To znamená, že daná kombinace by neměla být simulovaná, protože nepřinese nové informace o prohledávaném prostoru.

Funkce umožňuje zmenšit počet simulovaných kombinací. Novou skupinu kombinací z původní skupiny lze získat příkazem: `tS.p=new_combinations([~is_in_p],:)`

Práce s nastavením (properties)

V proměnné *properties* je uloženo nastavení pro danou metodu. Veškeré vlastnosti, které prohledávající metoda má nastavitelné, jsou uložené v této proměnné. O úpravu hodnot uživatelem se stará program MPE. Příklad použití, pokud se jedná o číselnou hodnotu:

```
>> properties{1,2};
ans =
        100
```

Pokud se jedná o textový řetězec, je vhodné použít funkci `strcmpi`, která porovnává textové řetězce a nebere ohled na velikost písmen (case-insensitive).

```
if 1==strcmpi(properties{2,2},'On')
    ...
end
```

Vyčistění grafu parametrů (draw_axes_p)

```
hObject.draw_axes_p;
drawnow
```

Vyčistí levý graf, ve kterém se nachází kombinace hledaných parametrů. Po vyčistění se znovu provede vykreslení čáry a hodnoty hranic intervalů a názvů hledaných parametrů. Protože je funkce součástí objektu MPE_GUI, je nutné mít při volání funkce odkaz (handle) na tento objekt. Funkce `m_solve` má odkaz na objekt pokaždé.

Vyčistění grafu průběhů simulací (draw_axes_y)

```
hObject.draw_axes_y;
drawnow
```

Vyčistí pravý graf, ve kterém se nachází průběhy simulací. Po vyčistění je graf prázdný. Protože je funkce součástí objektu MPE_GUI, je nutné mít při volání funkce odkaz (handle) na tento objekt. Funkce `m_solve` má odkaz na objekt pokaždé.

Vykreslení hodnoty chyby (draw_error)

```
hObject.draw_error(error_max,error,'k','LineWidth',3)
drawnow
```

Funkce `draw_error` zajištuje vykreslení hodnoty chyby do prvního intervalu v levém grafu. Kde *error_max* je horní hranice intervalu a *error* je hodnota chyby, kterou chceme vykreslit. Dále pak libovolný počet vstupů, které definují vlastnosti vykreslené čáry. Vlastnosti jsou stejné, jak u funkce `plot`. Protože je funkce součástí objektu MPE_GUI, je nutné mít při volání funkce odkaz (handle) na tento objekt. Funkce `m_solve` má odkaz na objekt pokaždé.

Vykreslení kombinace parametrů (draw_p)

```
hObject.draw_p(parameters,'b','LineWidth',1)
drawnow
```

Funkce `draw_p` vykresluje jednu nebo více kombinací parametrů do intervalů v levém grafu. Kde parametr *parameters* je matice parametrů. Řádky matice představují jednotlivé hledané parametry a konstanty. Konstanty se nevykreslí. Dále pak libovolný počet vstupů, které definují vlastnosti vykreslené čáry. Vlastnosti jsou stejné, jak u funkce `plot`. Protože je funkce součástí objektu MPE_GUI, je nutné mít při volání funkce odkaz (handle) na tento objekt. Funkce `m_solve` má odkaz na objekt pokaždé.

Vykreslení průběhu hledání

Pokud prohledávací metoda provádí simulovaní kombinací dávkově, tedy například po generacích, o vykreslení průběhu simulace se stará program MPE. Pokud se funkce `fitness_function` volá jen pro jednu kombinaci parametrů, je potřeba se o vykreslení průběhu hledaní postarat.

```
NumOfSimulated=length(hObject.hS.error);
TimeToEnd=tic;
tic
for i=1:NumOfSimulations
    %% fitnes fnc + search strategy
    ...
    %% progress bar
    if toc>1
        tic
```

```
            time1=(toc(TimeToEnd)/((length(hObject.hS.error)-NumOfSimulated),...
                /NumOfSimulations))-toc(TimeToEnd);
            time2=properties{4,2}-toc(TimeToEnd);
            set(hObject.FigureElements.Status.rectangle,'Position',...
                [0,0,(length(hObject.hS.error)-NumOfSimulated),...
                /NumOfSimulations,1]);
            set(hObject.FigureElements.Status.Text,'visible','on',...
                'string',[num2str(round(((length(hObject.hS.error)-,...
                NumOfSimulated)/NumOfSimulations)*100)),...
                '% ('num2str((length(hObject.hS.error)-NumOfSimulated)),'/',...
                num2str(NumOfSimulations)'),...
                'Time to end:',datestr(min(time1,time2)/86400, 'HH:MM:SS')]);
            drawnow
        end
    end
```

Jedná se o složitější případ, kdy kromě počtu simulací je zadána maximální doba simulace. Tato doba je uložena v tabulce `properties{4,2}` v sekundách. Algoritmus provede překreslení maximálně každou sekundu z důvodů zrychlení celého procesu. Doba do konce prohledávání se počítá z doby a počtu simulací, které už proběhly.

Užitečné typy

```
Intervals = hP.maxs(:)-hP.mins(:);
```

Stanoví velikost intervalů.

```
PositionOfParameters = find(Intervals>0);
```

Pozice, na kterých jsou parametry.

```
PositionOfConstant = find(Intervals==0);
```

Pozice, na kterých jsou konstanty.

```
Intervals = Intervals(PositionOfParameters);
```

Intervaly pouze parametrů.

```
Constants = Intervals(PositionOfConstant);
```

Intervaly pouze konstant.

B Definice problému pro oscilátor

```
clc; clear all; close all;
addpath(strcat(pwd,'/Funs'));
%% Vytvori objekt, ktery bude obsahovat cestu k datovim souborum
hD=MPE_D;
hD.set_D(strcat(pwd,'/Data/D1.mat'),0,1);    % decimace zadna, id 1
hD.set_D(strcat(pwd,'/Data/D4.mat'),0,4);    % decimace zadna, id 4
%% Vytvori objekt, ktery bude obsahovat cestu k modelum
hM=MPE_M;
hM.set_M(strcat(pwd,'/Models/M2_bez_tihove_sily.slx'),2); % id 2
hM.set_M(strcat(pwd,'/Models/M4_bez_treni_s_tihovou_silou'),4); % id 4
hM.table{2}.parameters  % vypise nalezene paremetry v modelech
%% Vytvori objekt, ktery bude obsahovat hranice parametru
hP=MPE_P;
hP.set_P('T',0,0.2);              %   T = 0.1      [N]
hP.set_P('b',0,1.8);              %   b = 0.9      [Ns/m]
hP.set_P('c',0,10);               %   c = 5        [-]
hP.set_P('dx0',0);                %   dx0 = 0      [m/s]
hP.set_P('k',0,20);               %   k = 10       [N/m]
hP.set_P('m',0.01,0.51);          %   m = 0.25     [kg]
hP.set_P('x0',0.5);               %   x0 = 0.5     [m]
%% Vytvori objekt, ktery bude obsahovat vysledky estimace
hS=MPE_S('MSE'); % Prumerna absolutni chyba (MAE,MSE,MPE,RMSE,NRMSE,PSNR)
hS.set_S(1,2,3.3e3);     % data D3, model M3, vaha 1000
hS.set_S(4,4,1e3);  % data D3, model M3, vaha 1000
%% Zavola estimaci
hMPE_Grid=MPE_GUI(hD,hM,hP,hS,'Grid','NumberOfSimulations',100000);

%hMPE_MC=MPE_GUI(hD,hM,hP,hS,'MC','NumberOfSimulations',100000);

%hMPE_GA=MPE_GUI(hD,hM,hP,hS,'GA_Matlab',    'Generations',50,...
%                                            'PopulationSize',2000);
```

C Definice problému pro škrtící klapku

```
clc; clear all; close all;
addpath(strcat(pwd,'/Funs'));
%% Vytvori objekt, ktery bude obsahovat cestu k datum
hD=MPE_D;
hD.set_D(strcat(pwd,'/Data/D1_fast.mat'),400);
hD.set_D(strcat(pwd,'/Data/D2_fast.mat'),400);
hD.set_D(strcat(pwd,'/Data/D3_faster.mat'),400);
%% Vytvori objekt, ktery bude obsahovat cestu k modelum
hM=MPE_M;
hM.set_M(strcat(pwd,'/Models/dyn_est_R2012b_phi0_D1.slx'));
hM.set_M(strcat(pwd,'/Models/dyn_est_R2012b_phi0_D2.slx'));
hM.set_M(strcat(pwd,'/Models/dyn_est_R2012b_phi0_D3.slx'));
hM.table{1}.parameters
%% Vytvori objekt, ktery bude obsahovat hranice parametru
hP=MPE_P;
hP.set_P('J',0.1,10);       %   J   [g*m^2]
hP.set_P('b',0.2,0.4);      %   b   [Nms/rad]
hP.set_P('f0',0.0864);      %   f0  [Nm]
hP.set_P('k',0.8,1);        %   k   [Nm/rad]
hP.set_P('k0',0.17,0.23);   %   k0  [Nm]
%% Vytvori objekt, ktery bude obsahovat vysledky estimace
hS=MPE_S('MSE'); % Prumerna absolutni chyba (MAE,MSE,MPE,RMSE,NRMSE,PSNR)
hS.set_S(1,1,1e6);
hS.set_S(2,2,1e6);
hS.set_S(3,3,1e6);
%% Zavola estimaci
hMPE_Grid=MPE_GUI(hD,hM,hP,hS,'Grid','NumberOfSimulations',10000);

%hMPE_MC=MPE_GUI(hD,hM,hP,hS,'MC','NumberOfSimulations',10000);

%hMPE_GA=MPE_GUI(hD,hM,hP,hS,'GA_Matlab',    'Generations',20,...
%                                            'PopulationSize',500);
```

D Definice problému pro DC motor

D.1 Definice pro program MPE

```
restoredefaultpath;
clc; clear all; close all;
addpath(strcat(pwd,'/Funs'));
%% Vytvori objekt, ktery bude obsahovat cestu k datum
hD=MPE_D;
hD.set_D(strcat(pwd,'/Data/D16_Without_flywheel.mat'),400);
hD.set_D(strcat(pwd,'/Data/D17_With_flywheel.mat'),400);
hD.set_D(strcat(pwd,'/Data/D18_Without_flywheel.mat'),400);
hD.set_D(strcat(pwd,'/Data/D19_With_flywheel.mat'),400);
hD.set_D(strcat(pwd,'/Data/D20_Without_flywheel.mat'));
%% Vytvori objekt, ktery bude obsahovat cestu k modelum
hM=MPE_M;
hM.set_M(strcat(pwd,'/Models/DC_without_flywheel.slx'));
hM.set_M(strcat(pwd,'/Models/DC_with_flywheel.slx'));
hM.table{1}.parameters
%% Vytvori objekt, ktery bude obsahovat hranice parametru
hP=MPE_P;
hP.set_P('J',0,30);          % J      [g.mm^2]
hP.set_P('Jz',81.977);       % Jz     [g.mm^2]
hP.set_P('Mz',0);            % Mz     [Nm]
hP.set_P('T',0,5e-3);        % Mz     [Nm]
hP.set_P('dphi0',0);         % dphi0  [rad/s]
hP.set_P('p1',0,30);
hP.set_P('p2',0,4000);
hP.set_P('phi0',0);          % phi0   [rad]
%% Vytvori objekt, ktery bude obsahovat vysledky estimace
hS=MPE_S('MSE');             % MAE, MSE, MPE, RMSE, NRMSE, PSNR
hS.set_S(3,1);
hS.set_S(4,2);
%% Zavola estimaci
hMPE_GUI=MPE_GUI(hD,hM,hP,hS,'GA_Matlab');
```

D.2 Definice pro program PE

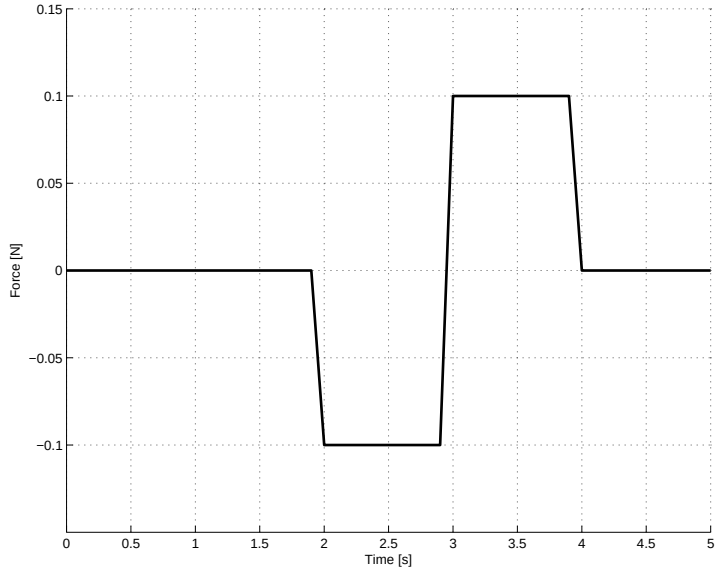
```
load D18_Without_flywheel_PES
p1 = 0; p2 = 0; p3 = 0; phi0 = 0; dphi0 = 0;
open_system('DC_without_flywheel_PES_v2')
data = [u.time, u.signals.values, phi.signals.values];
Exp = sdo.Experiment('DC_without_flywheel_PES_v2');
Exp.InputData = timeseries(data(:,2),data(:,1));
phiSig = Simulink.SimulationData.Signal;
phiSig.Name      = 'phi';
phiSig.BlockPath = 'DC_without_flywheel_PES_v2/Integrator2';
phiSig.PortType  = 'outport';
phiSig.PortIndex = 1;
phiSig.Values    = timeseries(data(:,3),data(:,1));
Exp.OutputData = phiSig;
phi0 = sdo.getParameterFromModel('DC_without_flywheel_PES_v2','phi0');
phi0.Value = 0;
phi0.Free  = false;
dphi0 = sdo.getParameterFromModel('DC_without_flywheel_PES_v2','dphi0');
dphi0.Value = 0;
dphi0.Free  = false;
Exp = sdo.Experiment('DC_without_flywheel_PES_v2');
Exp.InputData = timeseries(data(:,2),data(:,1));
phiSig.Values = timeseries(data(:,3),data(:,1));
Exp.OutputData = phiSig;
Exp.Parameters = [phi0;dphi0];
Simulator  = createSimulator(Exp(1));
Simulator  = sim(Simulator);
SimLog     = find(Simulator.LoggedData,...
get_param('DC_without_flywheel_PES_v2','SignalLoggingName'));
Phi(1) = find(SimLog,'phi');
p = sdo.getParameterFromModel('DC_without_flywheel_PES_v2',{'p1','p2','p3'});
p(1).Minimum = 0;
p(1).Maximum = inf;
p(2).Minimum = 0;
p(2).Maximum = inf;
p(3).Minimum = 0;
p(3).Maximum = inf;
```

```
s = getValuesToEstimate(Exp);
v = [p;s]
estFcn = @(v) DC_Objective(v,Exp);
opt = sdo.OptimizeOptions;
opt.Method = 'lsqnonlin';
vOpt = sdo.optimize(estFcn,v,opt)
Exp  = setEstimatedValues(Exp,vOpt);
Simulator  = createSimulator(Exp(1));
Simulator  = sim(Simulator);
SimLog     = find(Simulator.LoggedData,...
get_param('DC_without_flywheel_PES_v2','SignalLoggingName'));
Phi(1) = find(SimLog,'phi');
sdo.setValueInModel('DC_without_flywheel_PES_v2',vOpt);
bdclose('DC_without_flywheel_PES_v2')
```

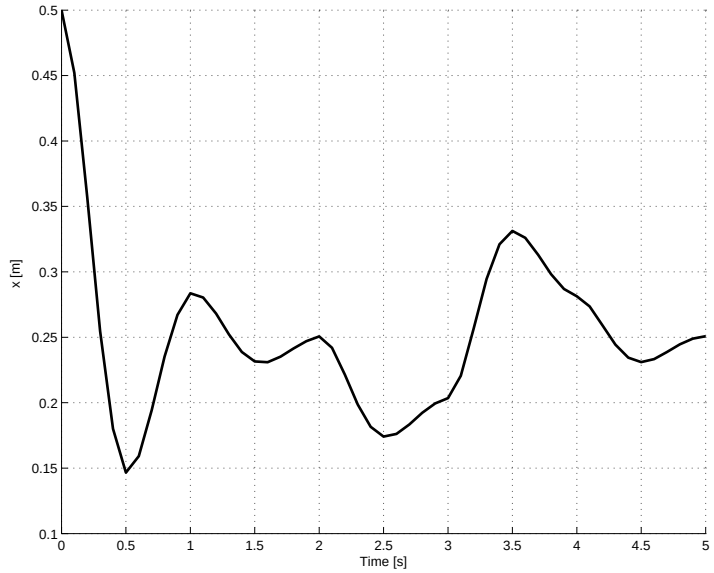
```
function vals = DC_Objective(v,Exp)
r = sdo.requirements.SignalTracking;
r.Type      = '==';
r.Method    = 'Residuals';
r.Normalize = 'off';
Exp  = setEstimatedValues(Exp,v);
Error = [];
for ct=1:numel(Exp)
Simulator = createSimulator(Exp(ct));
Simulator = sim(Simulator);
SimLog  = find(Simulator.LoggedData,...
get_param('DC_without_flywheel_PES_v2','SignalLoggingName'));
phi = find(SimLog,'phi');
phiError = evalRequirement(r,phi.Values,Exp(ct).OutputData(1).Values);
Error = [Error; phiError(:)];
end
vals.F = Error(:);
end
```

E Grafy pro ukázkové studie

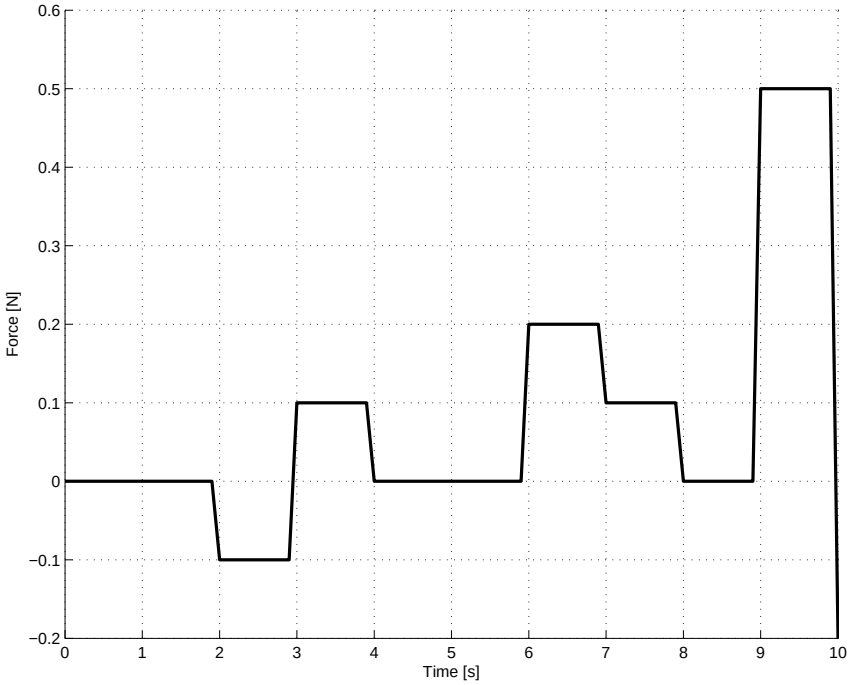
E.1 Oscilátor



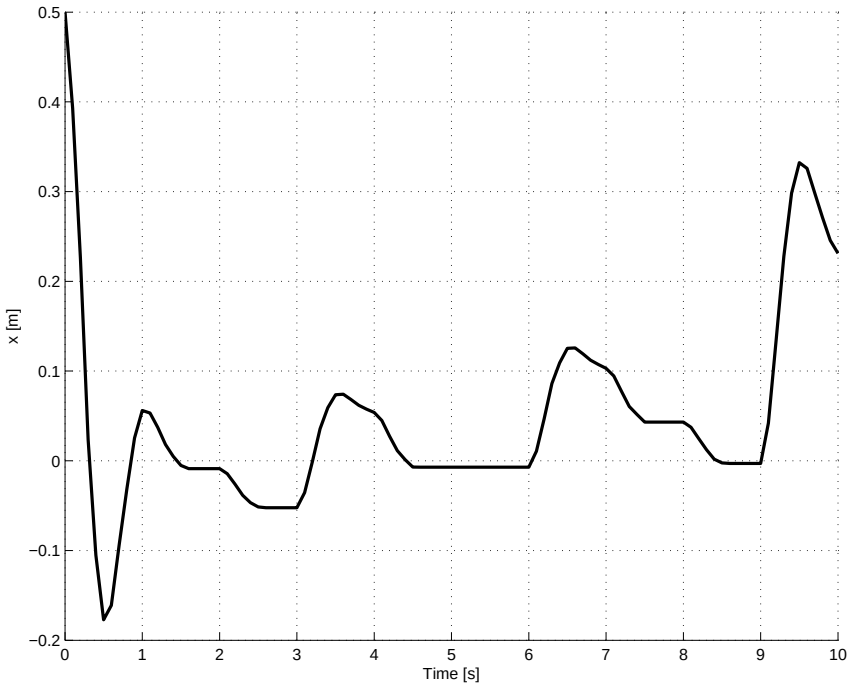
Obrázek 5.14: Vstupní síla prvního experimentu s vlivem gravitace a bez tření



Obrázek 5.15: Naměřená poloha prvního experimentu s vlivem gravitace a bez tření

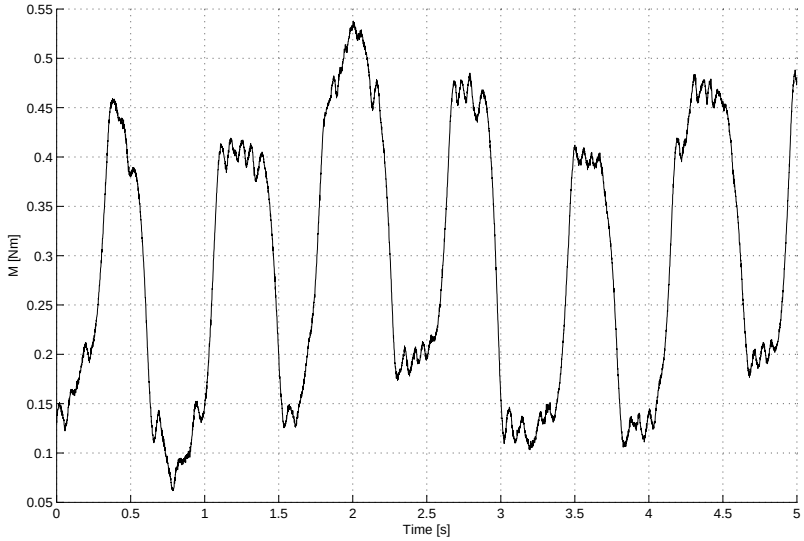


Obrázek 5.16: Vstupní síla prvního experimentu bez vlivu gravitace a se třením

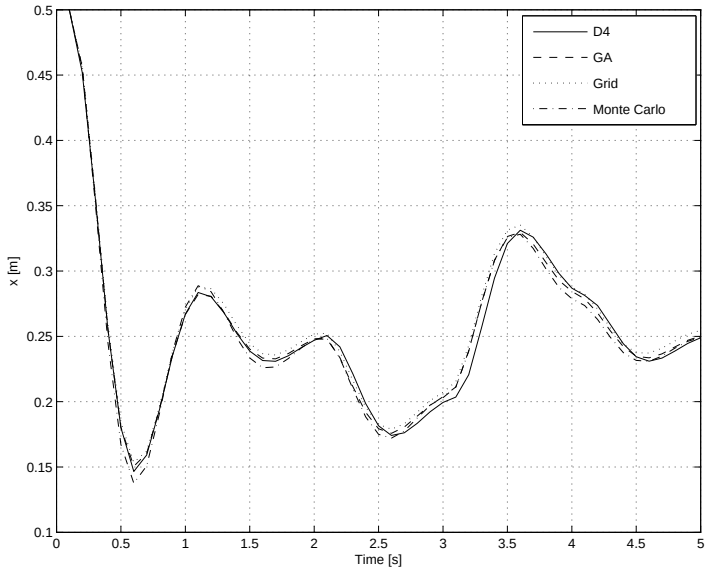


Obrázek 5.17: Naměřená poloha prvního experimentu bez vlivu gravitace a se třením

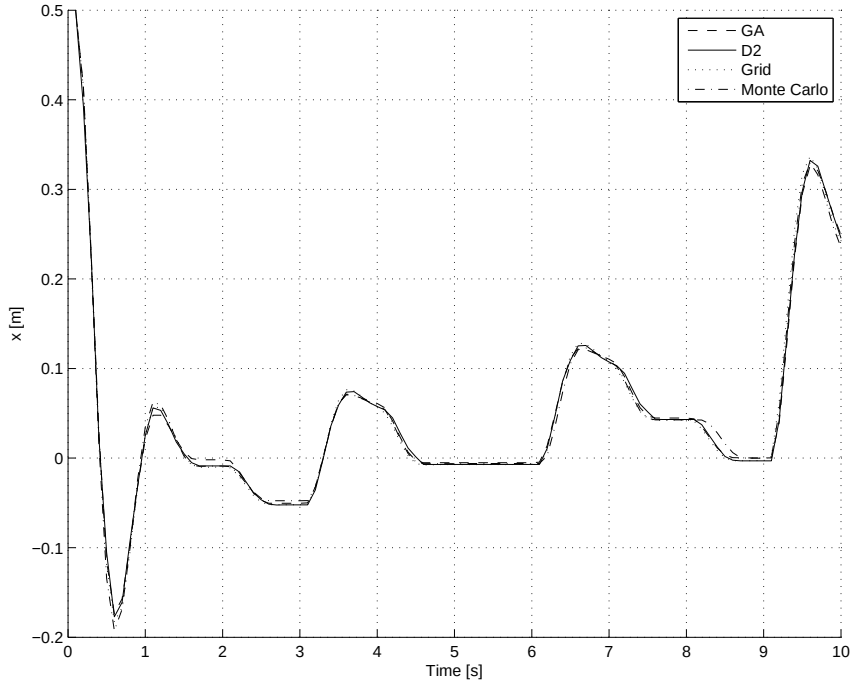
E.2 Škrtící klapka



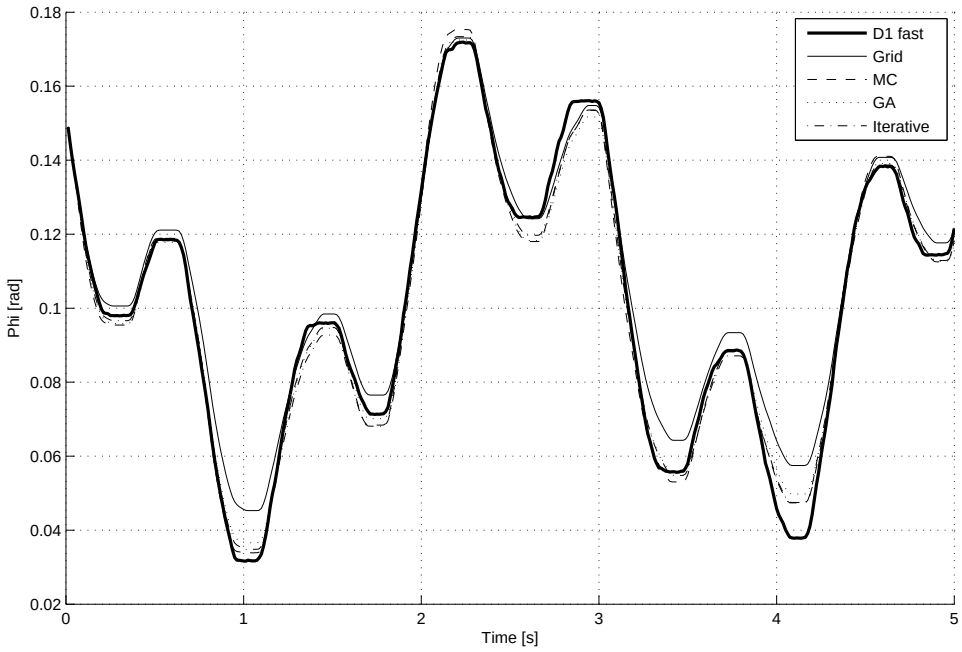
Obrázek 5.18: Vstupní signál pro škrtící klapku



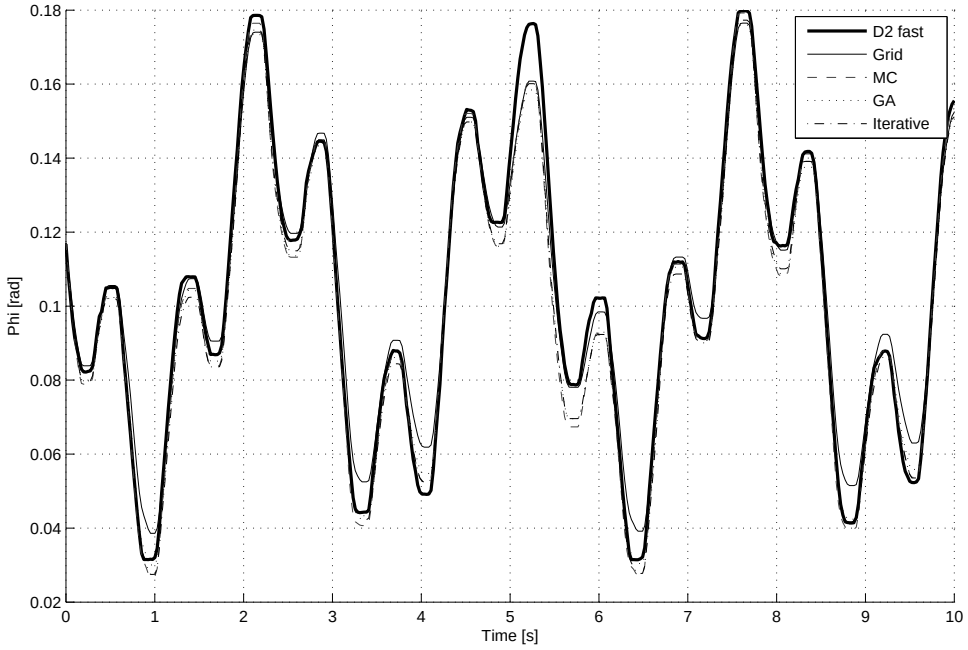
Obrázek 5.19: Odezva pro nalezené parametry pro první experiment



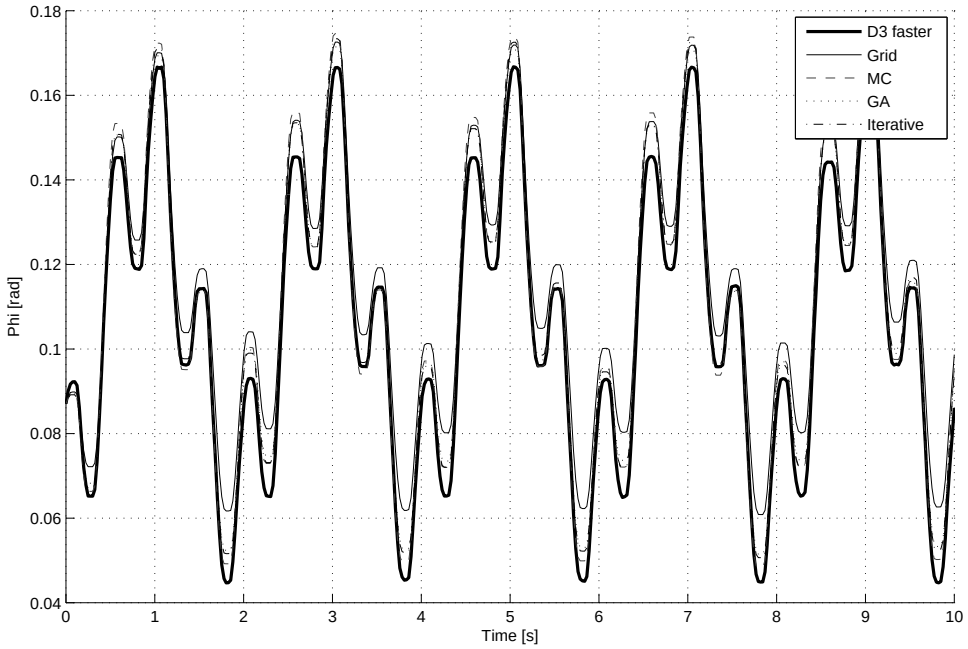
Obrázek 5.20: Odezva pro nalezené parametry pro druhý experiment



Obrázek 5.21: Odezva škrtící klapky



Obrázek 5.22: Odezva škrtící klapky



Obrázek 5.23: Odezva škrtící klapky