


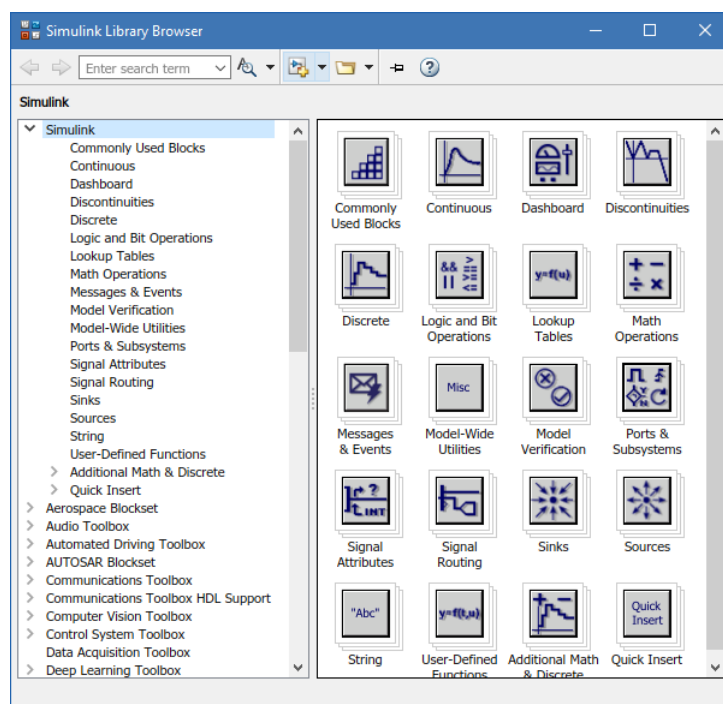


Školení Simulink I

MODELOVÁNÍ DYNAMICKÝCH SYSTÉMŮ

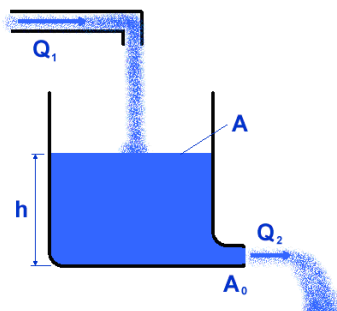
Úvod

- **Co je Simulink?**
 - nástroj pro simulaci dynamických systémů
 - spojité i diskrétní systémy
 - diferenciální/diferenční rovnice
 - modelování algoritmů
 - technické výpočty, řídicí systémy, zpracování signálu a obrazu, ...
- **Náplň školení**
 - od jednoduchého příkladu až po složitější modely
 - základní nastavení, tvorba blokových schémat, simulace, ...
- **Spuštění**
 - ikona  v záložce *HOME* (nebo příkaz *simulink* do *Command Window*)
 - zobrazí spouštěcí stánku Simulinku
 - založení nového modelu, šablony, nedávné modely, příklady
- **Otevření nového modelu**
 - *Blank Model*, ikona 
- **Simulink Library Browser**
 - ikona  v modelu (nebo příkaz *slLibraryBrowser* do *Command Window*)
 - knihovna Simulink
 - podknihovny – bloky
 - další nadstavby ... rozšíření Simulinku



⇒ Úloha 1:

→ Namodelujme si soustavu podle obrázku, kde Q_1 představuje přítok do nádoby, Q_2 odtok z nádoby, A je konstantní průřez nádoby a A_0 je průřez výtokového otvoru. Hledanou veličinou je parametr $h = ?$ výška hladiny kapaliny. *Rozměry nádoby:* průměr nádoby = 0,5 m $\Rightarrow A = 0.2 \text{ m}^2$, průměr otvoru = 0,05 m $\Rightarrow A_0 = 0.002 \text{ m}^2$. Přítok = $0.005 \text{ m}^3/\text{s}$.



- Dva přístupy v modelování:
 1. sestavit dif. rovnice matematicko-fyzikální analýzou
 2. využít bloky přenosů apod. pro odhad dynamiky – experimentálně nastavovat až po dosažení požadované odezvy (experimentální identifikace)
- Dle 1: Obecné ODE

$$\begin{aligned} \frac{dx(t)}{dt} &= F(x(t), u(t), t) & \text{zkráceně:} & \quad \dot{x} = F(x, u, t) \\ y(t) &= G(x(t), u(t), t) & & \quad y = G(x, u, t) \end{aligned}$$

- vysvětlení veličin: **u** ... vstup, **y** ... výstup, **x** ... stav, **t** ... čas
- čas se většinou explicitně neobjeví \Rightarrow časově invariantní systém

Potřebné rovnice pro sestavení modelu

1. Základní bilance: *změna akumulární veličiny = přítok – výtok*

Akumulární veličina se liší podle typu systému. Zde je to objem: $\frac{dV}{dt} = Q_1 - Q_2$

2. Dále platí:

$$V = A \cdot h \quad Q_2 = A_0 \sqrt{2gh}$$

- kde A_0 je průřez výtokového otvoru

3. Po dosazení rovnice 2. do 1. dostáváme

$$A \frac{dh}{dt} = Q_1 - A_0 \sqrt{2gh}$$

4. Osamostatníme derivaci na levé straně rovnice a konstantní parametry označíme k_1, k_2

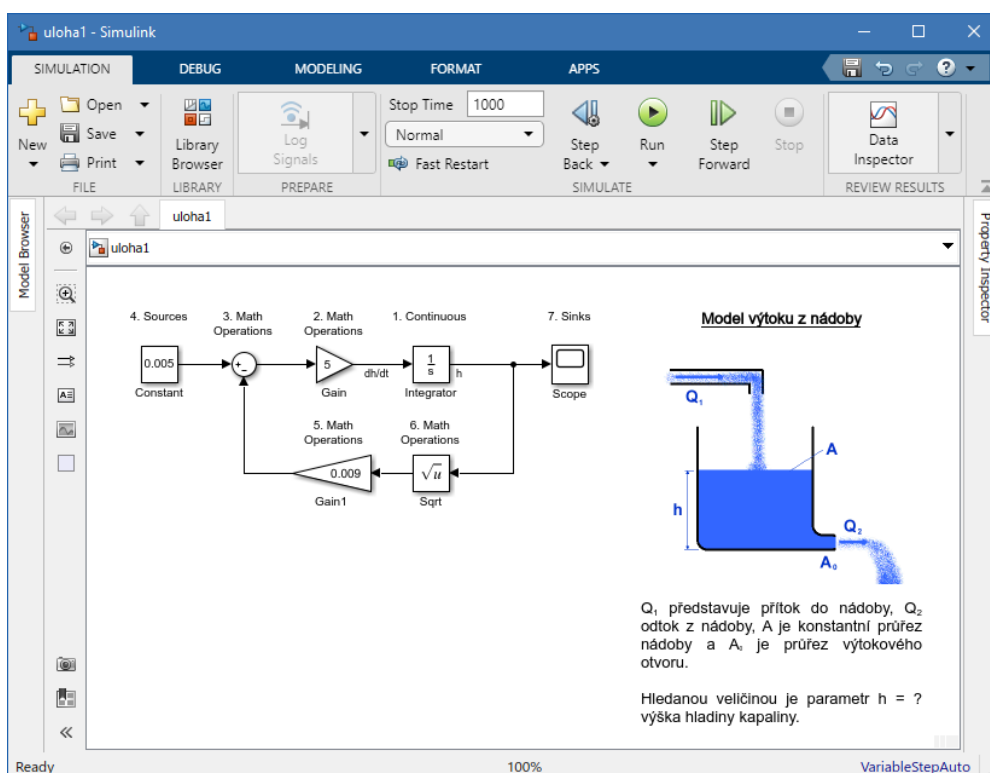
$$\frac{dh}{dt} = \frac{1}{A} (Q_1 - A_0 \sqrt{2g} \sqrt{h}) \quad k_1 = \frac{1}{A} \quad k_2 = A_0 \sqrt{2g}$$


5. Dostáváme výsledný vztah:
$$\frac{dh}{dt} = k_1(Q_1 - k_2\sqrt{h}) \quad \int \Rightarrow h$$

- Veličiny v úloze: $u \dots Q_1, y \dots h, x \dots h$

Sestavení schématu

- $A = 0.2 \text{ m}^2, A_0 = 0.002 \text{ m}^2, g = 9.81 \text{ ms}^{-2} \Rightarrow$ hodnota $k_1 = 5; k_2 = 0.009$;
- $Q_1 = 0.005 \text{ m}^3/\text{s}$
- nezajímá nás derivace h , ale pouze $h \Rightarrow$ proto musíme integrovat
- počet integrátorů odpovídá řádu systému



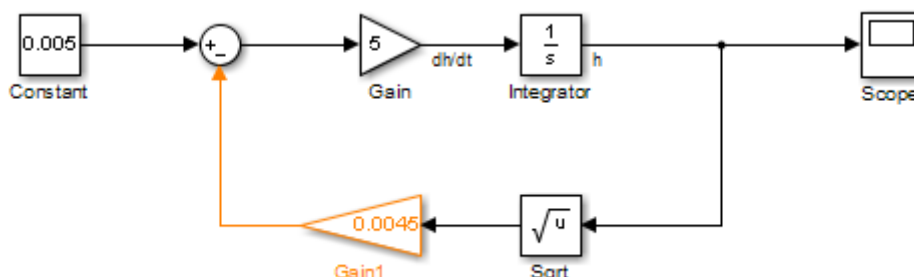
- **Postup:**
 - přidáváme bloky do modelu podle čísel v obrázku
 - uložení blokového schématu ... *SIMULATION* > *Save*
 - **soubor.slx** – XML soubor – může se používat diakritika
 - v minulosti **.mdl** – zde byl problém se znaky s diakritikou
 - editovat Sum ... dvojklik > změnit |++ na |+-
 - zapsání 5 do Gain ... dvojklik > zapsat 5 > OK
 - odbočení čáry do Scope:
 - stisknout pravé tl. na signálu > táhnout ke Scope
 - táhnout spojnicí od Scope k signálu h > napojí se
 - otevřít Scope ... dvojklik
 - spustit simulaci
 - doba simulace 1000 sec
 - tlačítkem Run ... 

- **Vysvětlení výsledku (co vidíme):**

- signál začíná v hodnotě 0, narůstá, ustálí se na hodnotě ... po ... sekundách.

⇒ Změna 1:

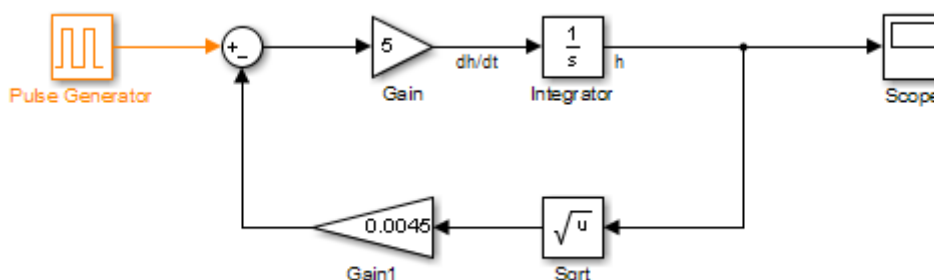
→ Co se stane, když přivřeme výtokový otvor na polovinu – sníží se A_0 a tedy koeficient k_2



- nastavit Gain1: 0.009 -> 0.0045
- simulovat – dojde u ustálení na vyšší hladině (potřeba vyšší tlak) a zpomalení soustavy (k_2 odpovídá nepřímo úměrně časové konstantě)

⇒ Změna 2:

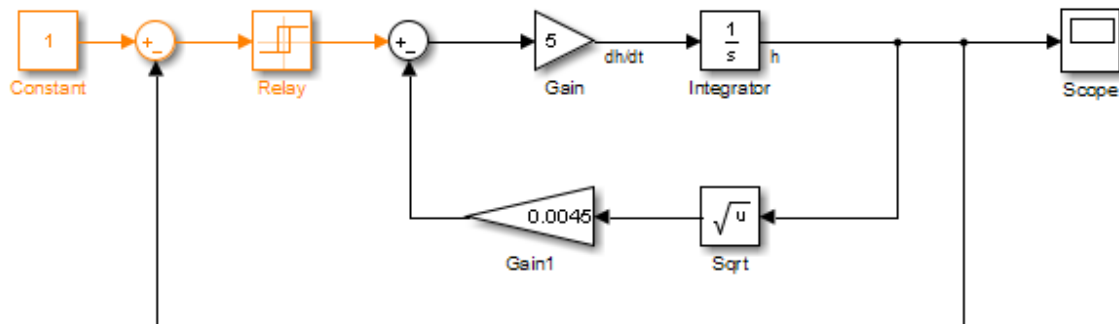
→ Jak docílíme, aby se vstup vypínal a zapínal, přičemž by se hladina ustalovala na: $h = 1 \pm 0.05$ m ?



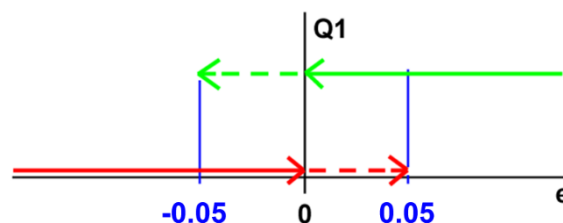
- Vstup máme zap – vyp ($0 - \max Q_1$): $h = 1 \pm 0.05$ m
 - zavedeme jiný vstup – místo Constant => Pulse Generator
 - amplituda = 0.005 ... 0.005 je plný přítok
 - šířka pulzu = 50% ... čím je pulz širší, tím více převládá přítok na výtokem, tj. hladina stoupá
 - perioda = 100 ... určuje rozkmit
 - postupně měníme šířku pulzu a periodu, opakovaně simulujeme, až se dostaneme na požadované hodnoty ... $p=100, w = 50\% > w = 75\% > w = 87\% > w = 93\%, w = 91\% > p=50 > w = 90\% > p=40$... OK
- **Závěr:** Cíl úlohy je udělat jednoduchý model, zkusit nastavení
 - nastavování parametrů na základě odezvy = experimentální identifikace
 - analyticky by bylo velice složité

⇒ Změna 3:

→ Jak docílíme, aby se hladina ustalovala na $h = 1 \pm 0.05$, tak jako v předchozí úloze, bez nutnosti manuálního hledání správného vstupního průběhu (metodou pokus-omyl) ?



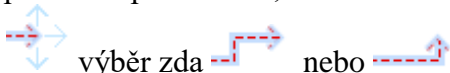


- Zpětná vazba, $h > h_0 \Rightarrow$ zavřít čerpání, $h < h_0 \Rightarrow$ spustit čerpání
 - odstranit vstup
 - přidat prvek Relay, Sum a Constant – zavést do Sum vstup h
 - Relay = model 2-polohového regulátoru
- **Princip:**
 - výsledek rozdílu hodnoty požadované výšky hladiny a skutečné výšky hladiny je regulační odchylka e
 - $e > 0$: hladina je níže než žádaná hodnota \Rightarrow napouštím
 - $e < 0$: hladina výše než žádaná hodnota \Rightarrow vypouštím
 - kolem hodnoty $e = 0$ by cyklovalo (nekonečně) rychle
 - \Rightarrow přidám hysterezi, aby se neopotřeboval akční člen
 - pozdržím napouštění do hodnoty $e = -0.05$
 - pozdržím vypouštění do hodnoty $e = 0.05$




- $switch\ on = 0.05$, $switch\ off = -0.05$
 - na začátku napouštím: $output\ when\ off = 0$, $output\ when\ on = 0.005$
- Z důvodu implementace musí být $switch\ on > switch\ off$
 - proto někdy můžeme chtít $output\ when\ off$ větší než $output\ when\ on$

Další poznámky k modelování:

- **Vkládání bloků zadáním jména**
 - kliknout levým tlačítkem myši do plochy modelu
 - začít psát jméno bloku – zobrazení roletky s odpovídajícími bloky
 - zvolit blok pro vložení – Enter, kurzorové šipky pro posun výběru, výběr myši
- **Změna velikosti bloků** – uchopení bloku za bílé čtverečky v rozích (+ **Shift** ... zachovává poměr stran, + **Ctrl** ... symetricky ke středu)
- **Přesun a kopírování bloků**
 - přesun bloku: levým tl. myši, kurzorové šipky
 - signálové spojnice se táhnou s blokem – zobrazuje se průběžně kudy povedou
 - vodící čáry pro zarovnání
 - kopírování bloku: pravým tl. myši

- utržení bloku: + **Shift**
 - můžeme ponechat v modelu více přednastavených variant
 - **zakomentování bloků**
 - nepoužité části modelu se nemusí mazat – lze je zakomentovat
 - pravý klik > *Comment Out* ... vymazání
 - pravý klik > *Comment Through* ... nahrazení spojnicí
 - pravý klik > *Uncomment*... zrušit zakomentování
 - (vybrat blok > *BLOCK* > *Comment Out/Through, Uncomment*)
- **Výběr více bloků:** myší pomocí rámečku, **Shift** + klikání myší
 - celý výběr lze posouvat, kopírovat
 - zarovnání a synchronizace velikosti
 - pravý klik > *Arrange*
 - (*FORMAT* > *ALIGN, DISTRIBUTE, MATCH*)
- **Vedení signálů**
 - červená čárkovaná spojnice - nepřipojeno
 - lomená čára: při tažení na chvíli pustit a poté zvolit směr, kam bude čára v bodě přerušení pokračovat, zda rovně nebo kolmo
 -  výběr zda  nebo 
- **Navigace v modelu**
 - *zoom* – kolečko myši (lze vypnout v preferencích), tlačítko *Zoom*
 - *pan* – stisknutí kolečka myši a posun
 - *fit to view* – tlačítko *Fit to View*, klávesa mezerník
 - *Explorer Bar* – jako procházení složek v OS
 - *Model Browser* – stromová struktura modelu
- **Blok má jméno**
 - jméno je povinné a pro blok jedinečné
 - lze měnit
 - ve výchozím nastavení se automatická jména schovávají
 - *DEBUG* > *Information Overlays* > *Hide Automatic Block Names*
 - u konkrétního bloku lze zobrazit/schovat
 - pravý klik > *Format* > *Show Block Name* > *Auto/On/Off*
 - (vybrat blok > *FORMAT* > roletka *Auto Name* > *Auto/On/Off*)
- Signál může mít jméno: dvojklik na signál ... editační čtvereček pro zapsání jména
- **Komentář**
 - dvojklik kamkoli do plochy > *Create annotation*, tlačítko *Annotation*
 - toolbar a kontextové menu – formátování, font, vkládání obrázků, ...
 - spojení komentáře s konkrétním blokem
 - vytáhnout spojnicí od hrany komentáře na blok
 - komentář se posouvá s blokem
- **Formát písma**
 - pravý klik na blok > *Format* > *Font style*
 - (vybrat blok > *FORMAT* > *FONT*)
- **Barvy**
 - blok má barvu popředí – rámeček, text, výstupní signál
 - blok má barvu pozadí – pozadí bloku
 - pravý klik na blok > *Format* > barva z textové nabídky, další barvy *Custom*
 - barva pozadí modelu – pravý klik do plochy modelu > *Canvas Color*
 - (vybrat blok/bloky > *FORMAT*)

- **Area** – seskupení bloků
- **Undo** – lze vrátit změny ve struktuře modelu, ne změny parametrů
- **Ukládání modelu:** *SIMULATION* > *Save*, ikona v záhlaví toolstripu
 - do R2012a přípona .mdl – textový soubor s popisem modelu
 - od R2012b přípona .slx – XML formát
- **Otevírání modelu:** zadáním jména do Command Windows, graficky poklepáním
- **Tisk:** na tiskárně – lze zvolit co tisknout
- **Kopírování do schránky**
 - *FORMAT* > *Screenshot* > výběr formátu (Metafile/Bitmap)
- **Nastavení modelu**
 - *MODELING* > *Model Settings*, ikona 
 - nastavení řešiče, nastavení ukládání dat ze simulace, ...
- **Nastavení editoru**
 - *MODELING* > *Environment* > *Simulink Preferences*


Property Inspector

- *MODELING* > roletka *DESIGN* > *Property Inspector*
- Informační panel v okně modelu
- Zobrazení a editace parametrů aktuálně vybraného bloku, signálu, komentáře nebo vlastností modelu
- Přístup k parametrům bloků bez otevírání dialogových oken

Model Data Editor

- *MODELING* > roletka *DESIGN* > *Model Data Editor*
- Informační panel v okně modelu
- Zobrazení a editace datových vlastností In/Out portů, signálů, stavů, parametrů

Model Explorer

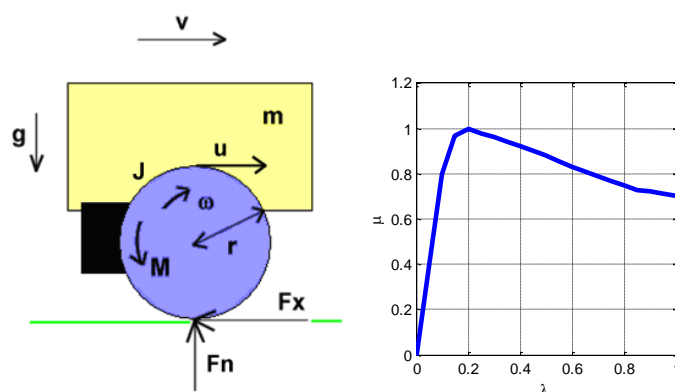
- *MODELING* > roletka *DESIGN* > *Model Explorer*, ikona 
- Zobrazení všech otevřených modelů
- Base Workspace
 - základní workspace MATLABu (ten, který je v okně MATLAB Desktop)
 - parametry (proměnné) viditelné pro všechny otevřené modely
- Model Workspace
 - přístup k vlastnímu workspace modelu
 - parametry modelu (proměnné) ukládané s modelem,
 - viditelné pouze pro daný model
- Správa konfigurací modelu
 - model může mít více konfigurací (nastavení řešiče, ...)
 - vždy jedna je aktivní – přepínání

Knihovny Simulinku

- Hierarchické uspořádání (*Simulink Library Browser*): bloky v knihovnách seřazeny podle jména
- Zobrazení knihoven tak jak jsou uloženy: pravý klik v browseru > *Open library*: bloky řazeny dle významu

⇒ Úloha 2:

Systém brzdění kola automobilu



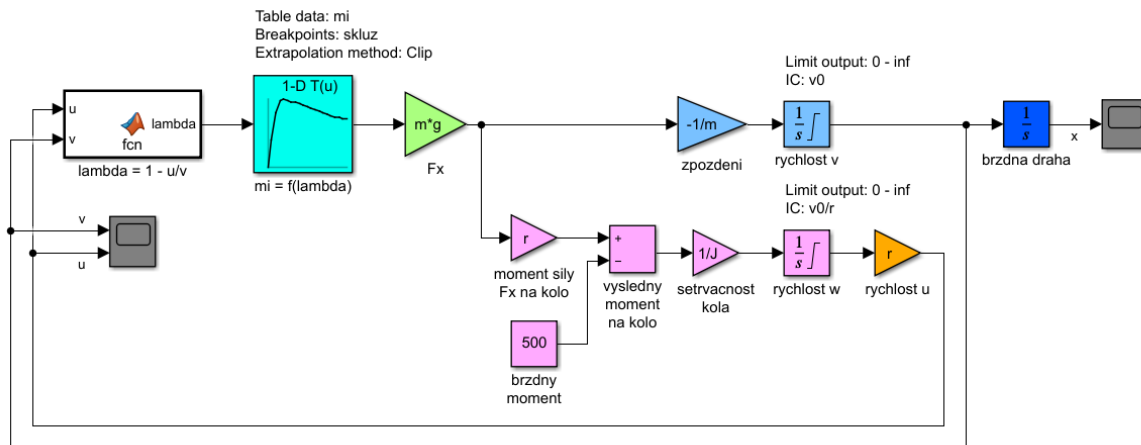
- Pohybové rovnice

$$\begin{aligned} m\dot{v} &= \sum F_i && \text{- suma sil působících ve směru pohybu} && \Rightarrow && m\dot{v} = -F_x \\ J\dot{\omega} &= \sum M_i && \text{- suma momentů působících na kolo} && \Rightarrow && J\dot{\omega} = rF_x - M \end{aligned}$$

$$F_x = \mu F_n \text{ - třecí síla, } F_n = mg \text{ - balance sil ve svislém směru} \Rightarrow F_x = \mu mg$$

$$\mu = f(\lambda) \text{ - koeficient tření je funkcí skluzu: } \lambda = \frac{v-u}{v} = 1 - \frac{u}{v}, \text{ kde } u = r\omega$$

$$\begin{aligned} \dot{v} &= -\frac{1}{m} F_x && \int \Rightarrow v && \lambda = 1 - \frac{u}{v} && \mu = f(\lambda) && F_x = \mu mg \\ \dot{\omega} &= \frac{1}{J} (rF_x - M) && \int \Rightarrow \omega && u &= r\omega \end{aligned}$$

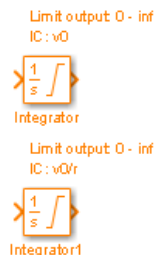


→ SPUSTIT parametry.m

```
%% Zadání parametrů pro model brzdění kola
m = 300; %kg
r = 0.25; %m
g = 9.81; %m/s2
J = 0.47; %kg*m2
v0 = 25; %m/s
% M = 500; %N*m

%% Tabulka hodnot funkce skluz - součinitel tření mi
skluz = 0:0.05:1;
mi = [0 0.4 0.8 0.97 1.0 0.98 0.96 0.94 0.92 0.9 0.88 0.855 0.83 0.81 0.79 0.77 0.75 0.73 0.72 0.71 0.7];
```

Continuous



- Základní knihovna pro dynamiku systémů
- Základní blok:
 - **Integrátor** (základní blok na tvorbu spojité dynamiky)
 - počáteční podmínka – interní: parametrem, externí: přivedeme další signál
 - reset – vnější signál, při splnění podmínky „natlačí“ poč. podmínku
 - podmínka resetování: hrana sestupná, vzestupná, obě nebo úroveň (log.1 = reset)
 - omezení integrace → od – do
 - saturation port – zda je či není v saturaci, přidá druhý výstup
 - tolerance – chyba numerické metody, která je přípustná: auto ... převezme globální nastavení řešiče
 - Second-Order Integrator – reset dx/dt při saturaci x
- Diferenciální rovnice s konstantními koeficienty:
 - stavový systém - spojitý systém můžeme linearizovat

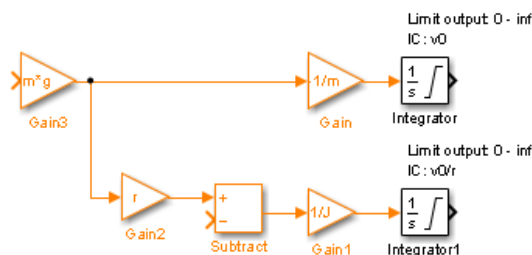
$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \quad \text{zkráceně:} \quad \begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

- *State Space*: zadání přímo matic A, B, C, D
- v přenosovém tvaru $y(s) = T(s)u(s)$, kde: $T(s) = \frac{b_m s^m + \dots + b_1 s + b_0}{a_n s^n + \dots + a_1 s + a_0}$
 - *Transfer Function*: zadání čitatele a jmenovatele
 - *Zero-Pole*: zadání pólů, nul a zesílení
- **Zpoždění**
 - *Transport Delay* – konstantní zpoždění
 - *Variable Time Delay* – nekonstantní zpoždění, vzhledem ke vzorku, který vystupuje – jak je zpožděný (starý)
 - *Variable Transport Delay* – nekonstantní zpoždění, vzhledem ke vzorku, který vstupuje – jak dlouho ho podržím
 - rozdíl je při nekonstantním zpoždění
- **PID regulace**
 - *PID Controller, PID Controller (2DOF)*
 - pokročilé bloky PID regulátorů
 - zapojení dle předvoleb – lze prohlédnout pomocí *Look Under Mask*
 - se *Simulink Control Design* automatické ladění
- **Derivace**
 - *Derivative*
 - **raději nepoužívat!!!**
 - Operace derivace = nekauzální operace – v čase t nelze vypočítat

$$\frac{dx(t)}{dt} = \lim_{\Delta t \rightarrow 0} \frac{x(t + \Delta t) - x(t)}{\Delta t}$$

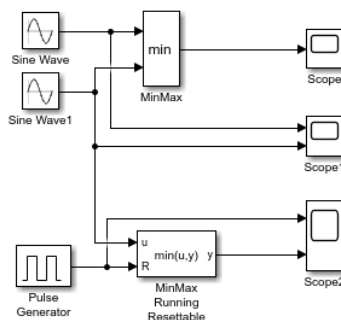
- v čase t bychom měli vědět i $t + \Delta t$, ale to nevíme
- v reálném výpočtu nahrazeno aproximací
 - pravý klik na blok > *Help for the Derivative block*
- vzorec aproximace – T_{previous} je předchozí krok simulace => výsledek závislý na konkrétní délce simulačního kroku
- výrazné zejména při skokovém vstupu
 - simulace nemůže rozlišit skok a rampu se strmostí jednoho kroku => derivace skoku je závislá na délce simulačního kroku (zejména patrné ve fixed step simulaci)
- v helpu je sekce, jak derivaci z modelu odstranit/nahradit
 - náhrada diskrétní diferencí
 - náhrada pomocí TF
 - odkaz Circuit Model – ukázka změny odvození modelu

Math Operations



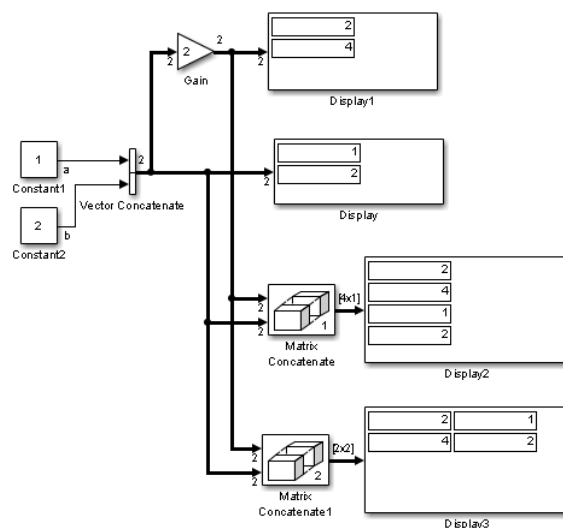
• Matematické operace

- základní matematické operace: *Sum*, *Add* (to samé, hranaté), *Subtract* (odečítání pro ty, co to nemusí měnit v *Sum* a *Add*), násobení, dělení
- elementární fce: *sign*, *abs*, ... *e^u* (schované další operace: *log*, *exp*, *pow*, ...)
- zaokrouhlovací funkce – *floor*, *fix*, *round*, *ceil*
- *Slider gain* - můžeme si graficky měnit konstantu
- hodnota polynomu – ze signálu *u* a parametrů [3 2 1] vypočte $3u^2 + 2u + 1$
- *maximum* a *minimum* – z vektorového signálu nebo v čase (s resetováním)



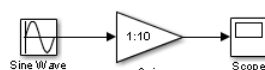
Pouziti_MinMax.slx

- trigonometrické fce: *sin*, *cos*, *tan*, *arcsin*, ... , *atan2* (2 vstupy pro zpracování číslo/0 ... vyhodí 90°)
- **Vektorové a maticové operace**
 - bloky pro zřetězení matic, permutaci dimenzí, *Squeeze* – odstranění singletonu (dimenze s rozměrem 1 ... 10x1x5 => 10x5), *Reshape* – změna tvaru matice
- **Vícerozměrné signály**
 - skalár – hodnota
 - vektor – více hodnot v jednom čase
 - matice – v každém čase 1 matice (obrázek v šedé škále)
 - 3-D pole – v každém čase více matic (obrázek v RGB – 3 matice)
 - zobrazení velikosti signálu:
 - *DEBUG > Information Overlays > Signal Dimensions, Nonscalar Signals*
 - vytváření vektorových signálů
 - vektory a matice v blocích *Sources*
 - zřetězení stávajících signálů



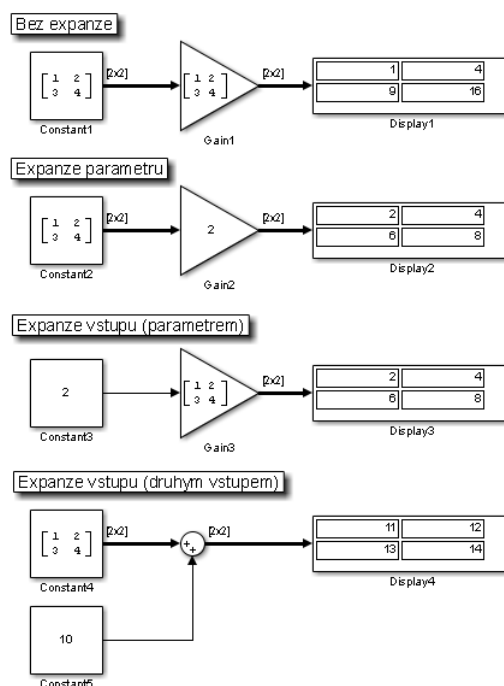
Pouziti_Zretezeni_signalu.slx

- skalární expanzí signálů



Pouziti_Expanze_SineWave.slx

- skalární expanze signálů



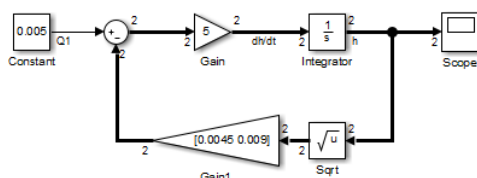
Pouziti_Expanze_moznosti.slx

- bez expanze - vektor šířky 2x2 jde do bloku zesílení a z něj jde opět signál šířky 2x2. Parametr: [1 2;3 4] ... první signál 1x, druhý 2x, ...

- skalární expanze parametru - vektor šířky 2x2 jde do bloku zesílení a z něj jde opět signál šířky 2x2. Parametr: 2 ... všechny signály 2x
- skalární expanze signálu (parametrem) - skalár jde do bloku zesílení a z něj jde signál šířky 2x2. Parametr: [1 2;3 4] ... signál násoben 1x, 2x, 3x a 4x
- skalární expanze signálu (druhým vstupem) - skalár jde do bloku sčítání a z něj jde signál šířky 2x2 dle druhého vstupu... ke všem prvkům je přičtena hodnota 10.

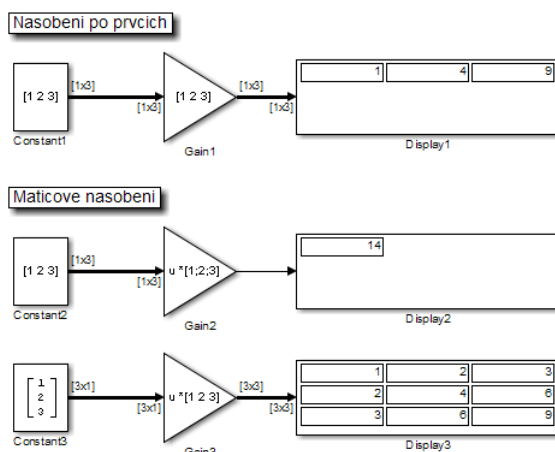
○ využití vektorového signálu:

- signály s různým významem zpracované zároveň
- většina bloků umí zpracovat vektorový signál
 - *Scope* – zobrazí obě veličiny
 - *Gain*, *Sqrt*, *Integrator* – aplikují se na všechny signály



Pouziti_Vektorove_signaly_integrace.slx

- maticové výpočty v Simulinku
 - v některých blocích volba, zda počítat maticově nebo po prvcích

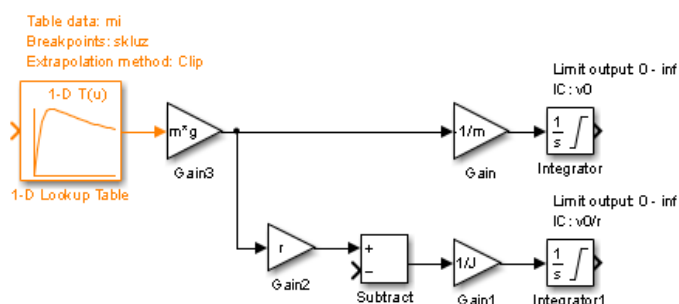


Pouziti_Maticove_vs_poprvcich.slx

- zjednodušení zobrazení (méně čar) – bloky Mux
- obvyklé oblasti využití maticových signálů
 - zpracování obrazu
 - zpracování vektorových signálů s historií (FFT)
 - samotný Simulink neumí (DSP System Toolbox)
- **Operace s komplexními čísly**

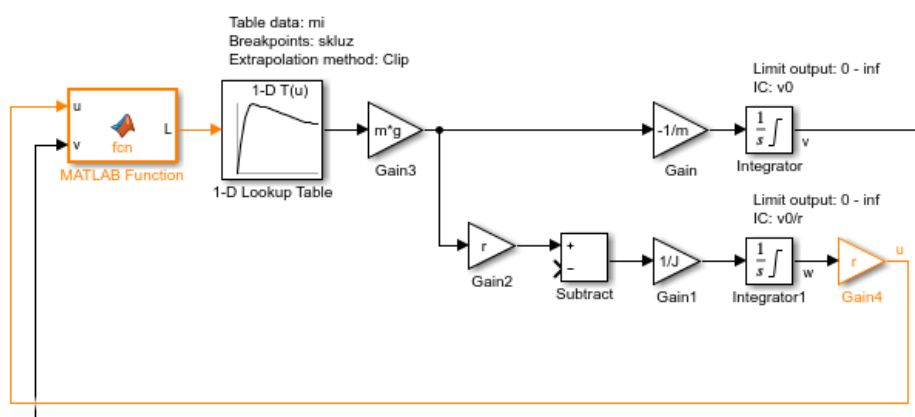
- Simulink má plnou podporu komplexních čísel
- bloky pro tvorbu a rozklad komplexních signálů

Lookup Tables



- **Funkce zadané tabulkou**
 - *1-D Lookup Table*
 - mezi hodnotami interpolace dle zvoleného algoritmu
 - mimo definovaný rozsah vstupu případná extrapolace
 - pozor na ni!
 - může způsobit neočekávané chování oproti realitě
 - nastavit clip, nebo dát saturaci před, či za *Lookup Table*
 - možnost grafické editace bodů, vykreslení, ...
 - *2-D Lookup Table* – plocha daná maticí, 2 vstupy
 - *n-D Lookup Table* – vícerozměrná matice
 - *Direct Lookup Table (n-D)* – bez interpolace, jen nejbližší hodnotu
- **Rozdělení do dvou kroků**
 - *Prelookup a Interpolation Using Prelookup* – nabízí více flexibility než předchozí, v základním použití je však stejné
- **Goniometrické funkce**
 - *Sine, Cosine* – zadané tabulkou s daným krokem, pro Real-Time aplikace, které nemají fci sin a cos, nebo pro simulaci tohoto typu náhrady

User-Defined Functions

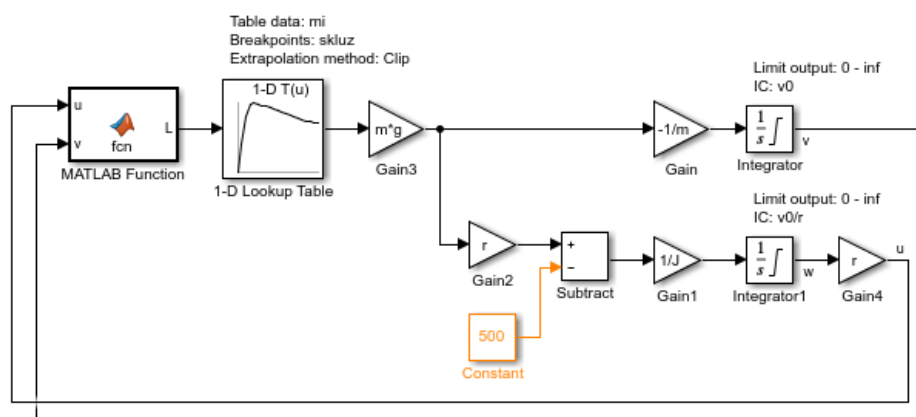


```
function L = fcn(u,v)
```

```
L = 1 - u/v;
```

- Zápis funkcí: liší se v obecnosti, obtížnosti vytvoření, účelu
- **Funkce napsaná v MATLABu**
 - *MATLAB Function*
 - otevře editor pro zápis funkce
 - funkce je uložena s modelem
 - počet vstupů a výstupů dán hlavičkou funkce
 - v editoru tlačítko *Edit Data* pro nastavení argumentů, zda se jedná o vstup/výstup ze Simulinku a nebo parametr z *Workspace*
 - kód je kompilován a spuštěn v binární podobě => rychlejší
 - lze využít jen příkazy a konstrukce podporující generování kódu, viz. Help (nepodporuje grafy, ...)
 - je podporováno pro generování kódu
 - pokud lze, je lepší využít toto
 - *Interpreted MATLAB Function*
 - odkaz na samostatnou MATLAB funkci
 - fun, fun(u), fun(u(1),u(2)), fun(u,p)
 - vstupní signál označen písmenem *u*
 - vstupem mohou být i parametry z *Workspace*
 - volá si MATLAB => pomalejší
 - není podporováno pro generování kódu
 - ukázka: Pouziti_InterpretedFunction.slx
 - *MATLAB System*
 - pro funkce s diskretním stavem
 - objektový zápis, uživatel implementuje dané metody
- **Funkce napsaná v C**
 - *C Caller, C Function*
- **Funkce napsaná (namodelovaná) z bloků Simulinku**
 - *Simulink Function, Function Caller*
- **S-funkce**
 - *S-Function, Level-2 MATLAB S-Function a S-Function Builder*
 - jazyky MATLAB, C/C++, Fortran
 - nejobecnější a nejsložitější na napsání
 - může obsahovat diskretní i spojitý stavy

Sources



- **Generování signálů**
 - Základní bloky: *Constant, Pulse Generator, Ramp, Step, Sine Wave*
 - *Signal Generator*: starší blok pro základní signály
 - *Repeating Sequence*:
 - periodicky se opakuje
 - průběh zadaný tabulkou (časy, hodnoty)
 - mezi časy lineární interpolace
 - *Signal Builder*:
 - grafický návrh průběhu, signál složený z přímek
 - lze definovat více signálů
 - přepínací záložky pro více skupin signálů – testování
 - náhodné signály:
 - *Chirp* – zrychlující se cosinusovka (pro frekvenční testy)
 - *Random Numer, Uniform Random Numer, Band-Limited White Noise*
 - hodiny:
 - *Clock* – pro vstup spojitého času
 - *Digital Clock* – pro vstup diskrétního času s danou periodou vzorkování
 - čítače: omezené, neomezené
- **Import dat do Simulinku**
 - *From Workspace*:
 - ve workspace MATLABu proměnné s daty: [čas, proměnná v čase]
 - po skončení extrapolace, opakování, nulování, poslední hodnota
 - v průběhu interpolace, držení hodnot do dalšího času
 - *From File* – načítání ze souboru, funkce stejné jako From Workspace

Pozn.: Předávání dat mezi MATLABem a Simulinkem

- do parametrů Simulinku lze zadávat proměnné v MATLABu
- vyhodnotí se při startu simulace

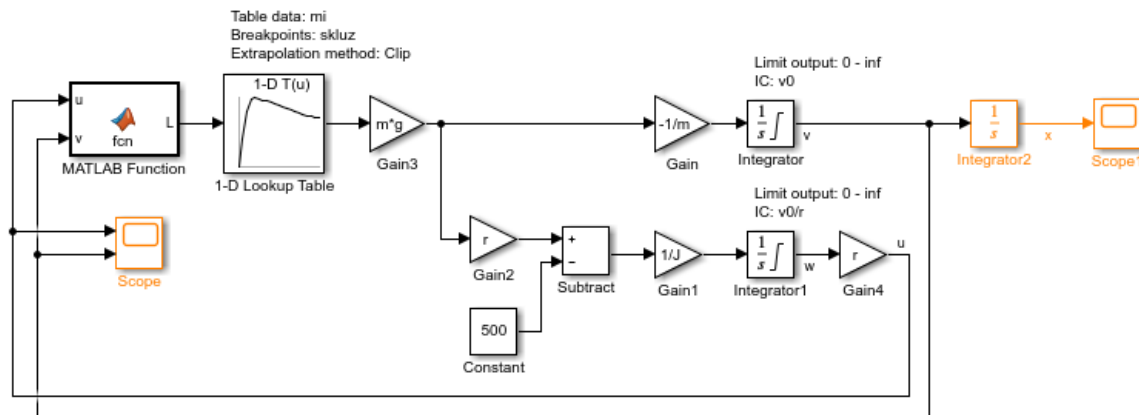
Pozn.: Signál vs. Parametr

- **signál** je to, co se při běhu simulace mění
- **parametr** by měl být neměnný
- pokud se má např. měnit hodnota vstupní konstanty, pak není dobré to modelovat změnou parametru bloku *Constant*, ale např. *Step*.



Pozn.: Čas v Simulinku



- obraz času reálného, běží různě rychle, běží nerovnoměrně rychle v závislosti na složitosti modelu (Simulink počítá stále naplno)
- nelze tedy odhadnout, jak dlouho bude simulace trvat
- nelze přímo propojit Simulink s reálným světem
 - k tomu jsou speciální nástroje (Simulink Desktop Real-Time)

Sinks

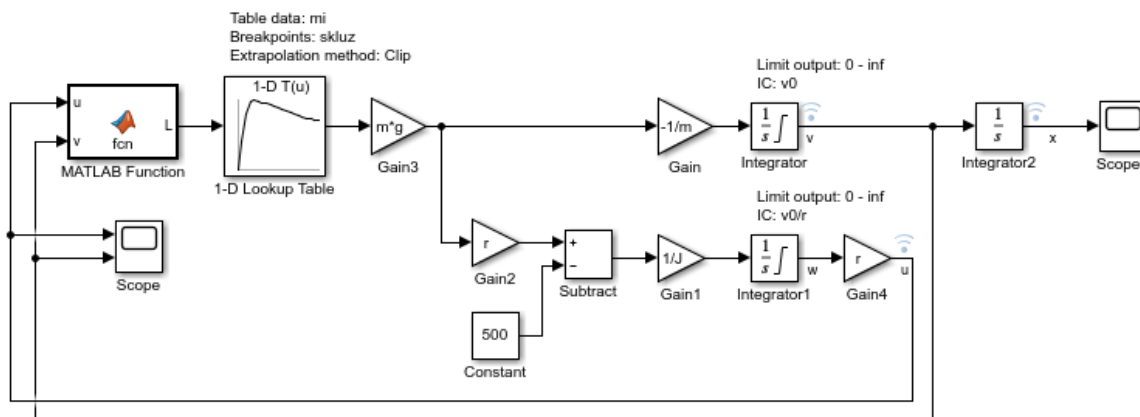



• Zobrazení

- *Display* – zobrazení okamžité číselné hodnoty signálu
- *Scope*
 - zobrazení dat v závislosti na čase
 - změny zobrazení:
 - lupy (x,y,x-y), autoscale (x,y,x-y) 
 - spouštění simulace
 - nastavení 
 - Main
 - *Numer of input ports* – pro zobrazení více signálů
 - *Sample time* – zobrazí se vzorky v časech daných periodou
 - *Axes scaling*
 - automatické (= za běhu simulace), manuální
 - autoscale na konci simulace
 - Time
 - *Time span* – Auto nebo ruční nastavení délky časové osy
 - Display
 - aktuální nastavení rozsahu Y-osy
 - manuální zadávání rozsahu Y-osy
 - Logging
 - délka dat držených v paměti – velká data zpomalují simulaci
 - *Decimation* – zobrazí jen každý n-tý vzorek
 - ukládání dat do proměnné v MATLABu
 - jméno proměnné ... **x**
 - formát uložení:
 - *Array* – matice času a hodnot, nejjednodušší
 - data: `x(:,1)`, `x(:,2)`
 - `plot(x(:,1),x(:,2))`
 - *Structure with time* – pro složitější signály
 - data: `x.time`, `x.signals.values`
 - `plot(x.time,x.signals.values)`
 - *Structure* – totéž, ale bez času
 - *Dataset* – nejkompaktnější
 - `plot(x)` ... vlastní vizualizace
 - `x{1}.Values` ... objekt *TimeSeries*
 - data: `x{1}.Values.Time`, `x{1}.Values.Data`

- **pozn.** před R2017a: `x.get(1).Values`
 - pro samostatné ukládání blok *To Workspace*
- Style  >  – nastavení čar a pozadí grafu
- Vykreslení grafu ze *Scope* ve standardním okně Figure v MATLABu
 - *File > Print to Figure*
 - možnost úpravy grafu, přidání popisků, uložení grafu, ...
- *XY Graph* – nezávisle proměnná, závisle proměnná
- Alternativa k blokům – viewers: pravý klik na signál > Create & Connect Viewer > ...
- **Export ze Simulinku**
 - *To Workspace*
 - jméno proměnné v MATLABu
 - max. délka – kolik posledních vzorků má uložit
 - Decimation – ukládá se každý n-tý vzorek
 - Sample time – ukládá vzorky v časech daných periodou
 - formát ukládání
 - *Array* – vektor hodnot
 - na rozdíl od *Scope* ukládá pouze signál bez času
 - vektor s časem je třeba uložit zvlášť
 - *Structure, Structure with time* – struktura
 - data: `x.time`, `x.signals.values`
 - *TimeSeries*
 - data: `x.Time`, `x.Data`
 - *To File* – přímý zápis do souboru





Simulation Data Inspector



- Grafické rozhraní pro zobrazení signálů z jednotlivých běhů simulace
 - otevření *SIMULATION > Data Inspector* 
 - určen zejména pro práci se signály logovanými do *Workspace*
- Zobrazení signálů v Simulation Data Inspectoru
 - nastavení logování signálu – u signálu se zobrazí modrá ikona logování
 - označit signál > *SIMULATION > Log Signals*
 - (pravý klik na signál > *Log Selected Signals*)
 - logované signály jsou automaticky přenášeny do Simulation Data Inspectoru
 - spustit simulaci => signály se objeví v Simulation Data Inspectoru
 - zaškrtnutí signálu = zobrazení v aktivním grafu
 - předchozí běh se přesune do sekce *Archive*

- Záložky Simulation Data Inspectoru
 - *Inspect* – zobrazení signálů, i více najednou
 - *Compare* – porovnání signálu mezi dvěma běhy, včetně grafu odchylek
- Export zobrazeného grafu do okna Figure – Send to Figure
- Export dat ze Simulation Data Inspectoru do *Workspace*
 - pravý klik na signál v seznamu či grafu > *Export*
 - (ikona *Export* v bočním menu)
- Zobrazení dat ukládaných pomocí *Scope*, *To Workspace*, *Out* nebo ukládaných stavů
 - *SIMULATION* > roletka *PREPARE* > *Configure Logging* > ☒ *Record logged workspace data in Simulation Data Inspector*

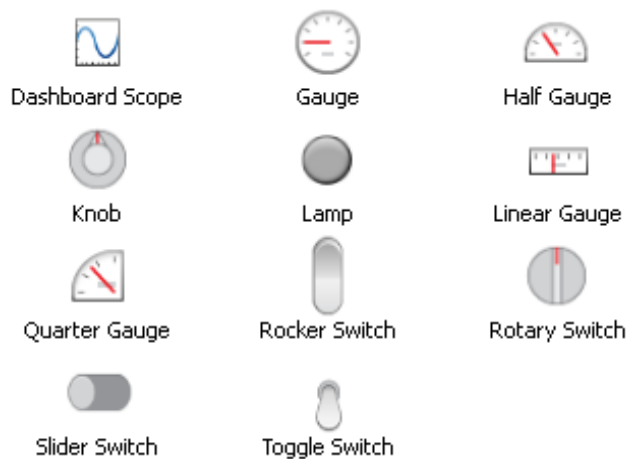
Pozn.: Starší verze před R2017a:

- přenesení logovaných signálů do Simulation Data Inspectoru nutné zapnout samostatně  >  *Send Logged Workspace Data to Data Inspector*
- Streamování signálu do Simulation Data Inspectoru bez jejich logování  >  *Stream Selected Signals to Data Inspector*

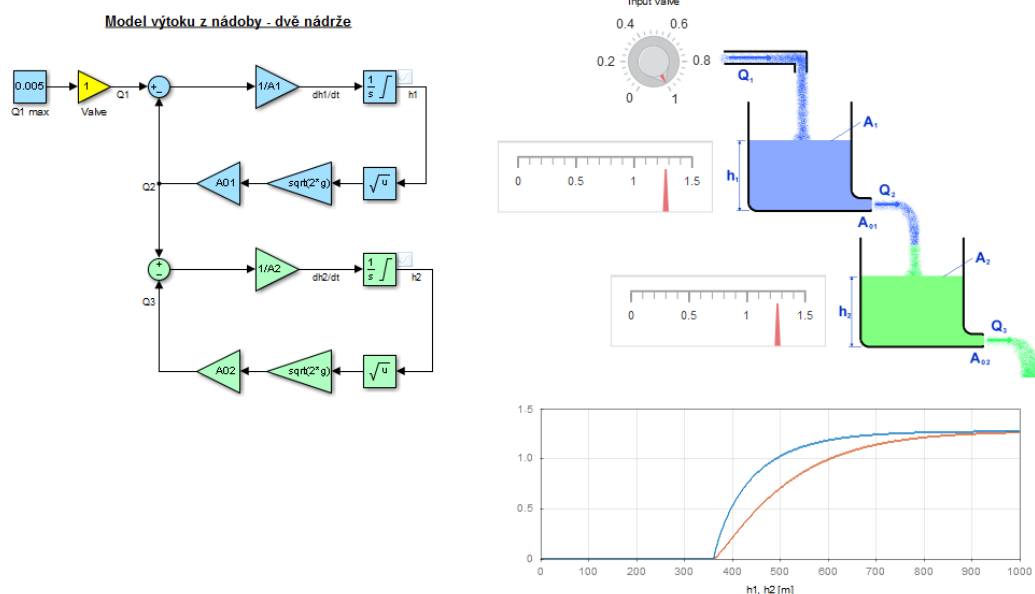
⇒ Úloha

→ Odsimulujte model s nastavením konstanty na 500 a 750 Nm. Výsledky zhodnoťte v nástroji Simulation Data Inspector.

Dashboard



- Interaktivní grafické prvky pro zadávání vstupu a vizualizaci výstupu

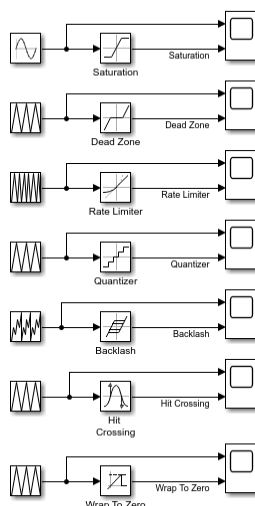


Pouziti_Dashboard.slx

Logic and Bit Operations

- **Logical Operator :**
 - AND, OR, NAND, NOR, XOR, NXOR, NOT,
- **Relační operátory:**
 - *Relational Operator, Compare to Constant , Compare to Zero*
 - porovnání dvou signálů nebo signálu s číslem – výstup boolean
 - pozor na $=$ a \neq ... dva double se jen tak nerovnejí (např. porovnání s 0 – signál nabude hodnotu -0,001 a v dalším kroku 0,001, tedy není = 0 nikdy!)
 - *Interval Test:* vhodná náhrada pro porovnání signálu s konstantou
 - *Combinatorial Logic:* matice výstupu kombinační tabulky, vstup se předpokládá 00,01,10,11
- **Bitové operace** – na bitech vnitřní reprezentace bitového čísla (shiftování), pro celočíselné signály
- **Edge Detection** – detekce, zda signál roste, klesá ... z kombinace současné a minulé hodnoty

Discontinuities

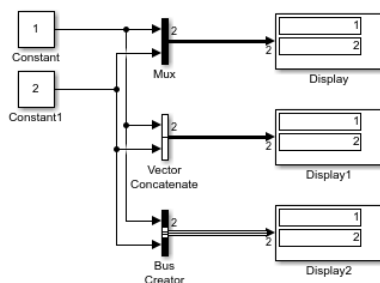


Pouziti_Discontinuities.slx

- **Typické nelineární prvky**, které lze těžko vyjádřit matematickým popisem, ale často se vyskytují
 - *Relay* – relé – již jsme použili
 - *Saturation* – nasycení (saturace)
 - *Dead Zone* – necitlivost
 - *Rate Limiter* – max. strmost signálu
 - pro všechny tři - dány pevně číslem nebo dynamicky (dalším signálem)
 - *Quantizer* – kvantizace signálu v úrovni (nikoli v čase)
 - *Backlash* – model „vůle“ (v převodech, ...)
 - *Hit Crossing* – detekce průchodu signálu danou úrovní
 - výstup log 0 a 1
 - lze zapnout algoritmus zero-crossing detection
 - zjemňuje krok pro nalezení přesného časového okamžiku protnutí
 - *Wrap To Zero* – vynulování moc velkého signálu
 - *Coulomb & Viscous fiction* - model tření

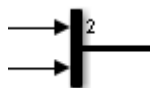
Signal Routing

- Nezpracovávají signál, ale jen ho posílají odněkud někam, spojují, rozčleňují
- **Spojení signálů**

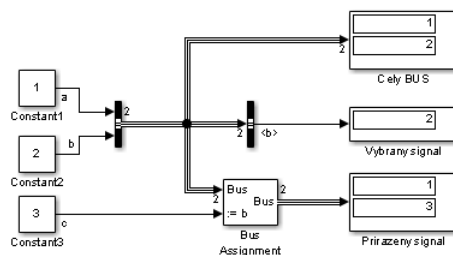


Pouziti_Mux_VC_BUS.slx

- *Mux, Demux* – slučování (multiplexování) signálů
 - skalární (a vektorové) signály sloučí do vektoru

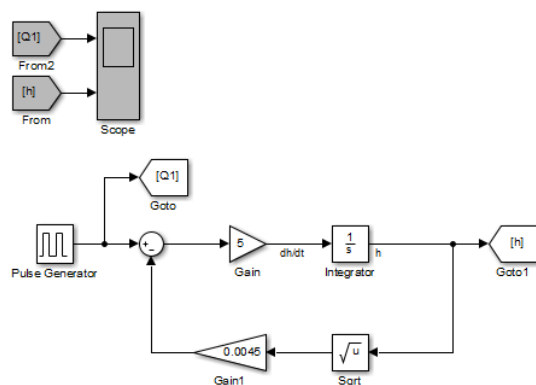


- vstupní signály musí mít stejný datový typ
- grafické odlišení vektorového signálu (BOLD): *DEBUG > Information Overlays > Nonscalar Signals, Signal Dimensions*
- nejčastěji pro
 - zjednodušení počtu čar v modelu
 - zobrazení více signálů v jednom grafu
- **Pozn.:** Blok *Mux* funguje podobně jako blok *Vector Concatenate*. Zatímco však *Vector Concatenate* skutečně vytváří nový vektor uložený v souvislé paměti, *Mux* slučuje signály pouze virtuálně (graficky v modelu) a žádný nový vektor se v paměti nevytváří.
 - (též při generování C kódu není v případě virtuálního vektoru (*Mux*) vytvářena žádná nová proměnná)
 - do *Mux* často dávám nesouvisející signály, čistě z důvodu přehlednosti, zobrazení, atd. – *Vector Concatenate* obvykle použiji, když vektor má funkční smysl pro model
- *Demux* opětovný pro rozklad na skaláry
- *Bus Creator, Bus Selektor* – vytváří „sběrnice“, signály definované jménem
 - obdoba *Mux* a *Demux*
 - na rozdíl od vektorového signálu sběrnice má všechny signály pojmenované a může slučovat signály různých datových typů



Pouziti_BUS.slx

- Bus Selektor vybírá signály dle jména – nezáleží na pořadí
- zejména pro složitější modely
- *Bus Assignment* – změna hodnoty jednoho signálu ve sběrnici (bez rozpojení)
- **Přepínače**
 - *Switch* - přepínač dle podmínky
 - *Multiport Switch* - přepínač na základě řídicího signálu
 - *Manual Switch* - pro manuální kliknutí – dvojklik = přepnutí
- **Přenos signálu jinam**
 - *Goto, From*



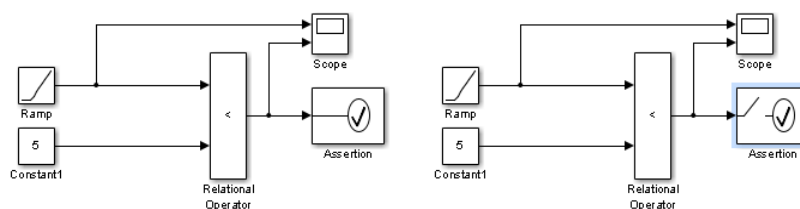
Pouziti_GotoFrom.slx

- Tag ... jméno, ze kterého Goto do kterého From se má signál přenést
- z jednoho Goto lze do více From (signály lze přece větvit)
- viditelnost: local (v dané vrstvě modelu), global (všude), scoped (dle bloku Goto Tag Visibility)
- *Goto Tag Visibility* – pro hierarchické modely
- *Data Store Read/Write/Memory* – jako Goto a From, ale se zápisem do paměti

Signal Attributes

- **Datové typy v Simulinku**
 - nastavení v blocích nebo blok *Data Type Conversion*
 - implicitní formát double – floating point 64-bit
 - další datové typy:
 - celočíselné: int8, int16, int32, int64, uint8, uint16, uint32, uint64
 - binární: boolean
 - floatové: single (32-bit)
 - fixed-point (vyžaduje Fixed-Point Designer)
 - pokud není důvod, je dobré nechat double (optimalizované, nejrychlejší)
 - důvody: zpracování obrazu, kvůli velikosti, pro simulaci chování např. filtru, který bude na reálném zařízení 8-bitový

Model Verification



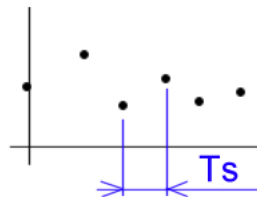
Pouziti_VaV_Assert.slx

- Ověření funkčnosti modelu
- **Základní blok:**
 - *Assertion*
 - vstup log.1, nedělá nic

- vstup log.0 – vyvolá akci: warning s výpisem času kdy nastalo, error - zastavení simulace s výpisem času kdy nastalo, definovaný callback
- vypínání – na 2 úrovních: lokálně každý blok zvlášť, globálně pro všechny blok v nastavení simulace
- Ostatní bloky jsou předdefinované podmínky kombinované s blokem Assertion

Diskrétní systémy

- Základním parametrem je perioda vzorkování – hodnota signálu existuje jen v těchto okamžicích



- Diskrétní čas

$$k = \frac{t}{T_s}$$

$$t = k \cdot T_s$$

- Diferenční rovnice

$$x_{k+1} = F(x_k, u_k, k)$$

$$y_k = G(x_k, u_k, k)$$

- Získána diskretizací diferenciálních rovnic, např.

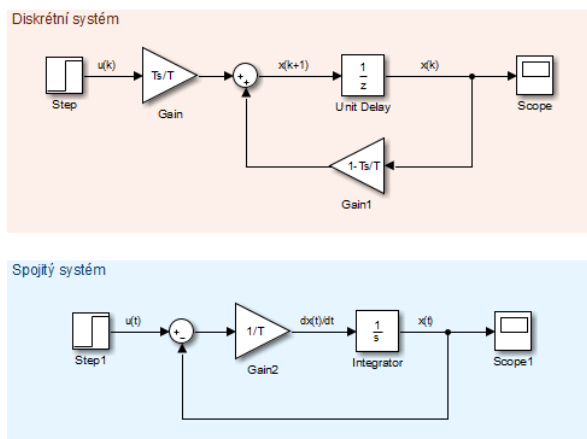
$$x(t) = x_k, \quad u(t) = u_k, \quad y(t) = y_k$$

$$\frac{dx(t)}{dt} \approx \frac{x_{k+1} - x_k}{T_s}$$

- koeficienty jsou závislé na T_s

Discrete

- Perioda vzorkování
 - zadaná – číslo
 - -1 ... přebírá periodu vzorkování ze vstupního signálu
- **Základní blok diferenční rovnice**
 - *Unit Delay*
 - zpoždění o jednu periodu vzorkování – definována v bloku
 - má počáteční podmínku



Pouziti_Discrete.slx

- **Složitější diferenční rovnice (v symbolice Z-transformace)**

- diskrétní náhrada integrace, diskrétní náhrada derivace, difference
- diskrétní systémy:

- diskrétní stavový prostor

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k + Du_k$$

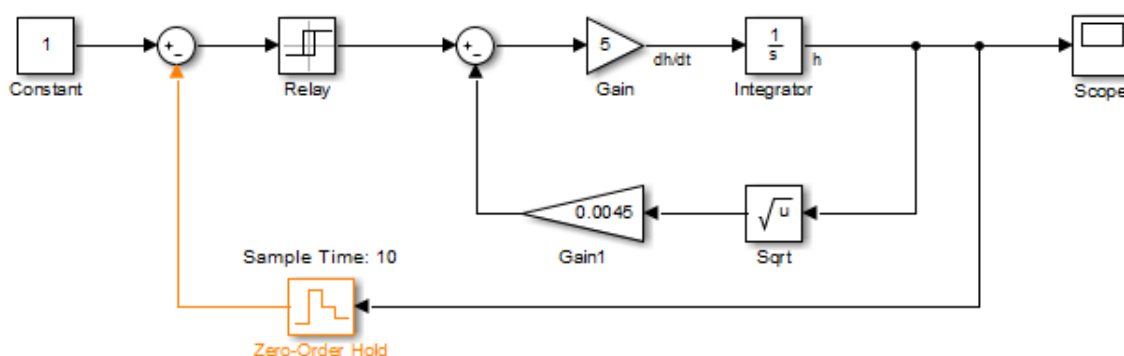
- diskrétní přenosová funkce, diskrétní zero-pole

$$y(z) = T(z)u(z), \text{ kde: } T(z) = \frac{b_m z^m + \dots + b_1 z + b_0}{a_n z^n + \dots + a_1 z + a_0}$$


- Diskrétní PID regulátory

- **Získání diskrétního signálu ze spojitého signálu**

- *Zero-Order Hold* – „drží poslední hodnotu“ po dobu periody vzorkování
 - má zadánu periodu vzorkování, kterou přeberou bloky za ním s -1, funkčně však signál neovlivňuje
 - nepoužívat pro změnu periody vzorkování z jednoho diskrétního signálu na druhý, k tomu slouží blok *Rate Transition* (knihovna *Signal Attributes*)



- Kombinace diskretních a spojitých bloků v modelu
 - za diskretní bloky lze přímo zapojit bloky spojitě
 - diskretní bloky mají na svém výstupu hodnotu z poslední periody vzorkování a spojitý blok si ji může kdykoli „přečíst“ i mezi periodami vzorkování
- Zobrazení periody vzorkování v modelu

- *DEBUG > Information Overlays > sekce Sample Time*
- ikona 
- *Memory* – zpoždění o jeden krok řešiče ... může sloužit při odstranění algebraických smyček








Simulace


- Pohled na simulaci jako na proces
- Simulace má
 - počáteční a cílový čas – *SIMULATION > Stop Time*, poč. čas v nastavení
 - počáteční podmínky a poč. čase – dané v blocích (Integrátor, State Space)
- Simulaci lze
 - proběhnout jednorázově
 - zpomalit – roletka *Run > Simulation Pacing*
 - krokovat – dopředu, zpět

Ovládání simulace

- záložka *SIMULATION*
 - ovládání simulace – spouštění, krokování

Krokování simulace

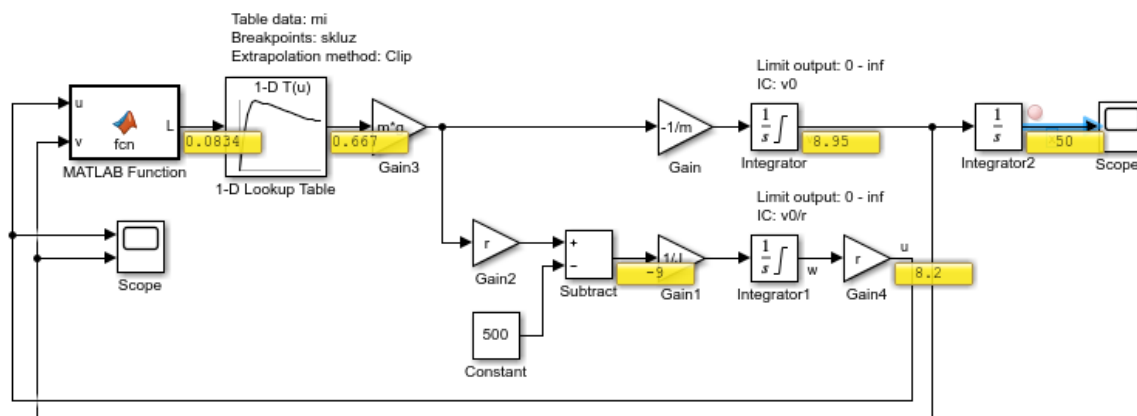
- Simulaci je možné začít krokovat od začátku – tlačítko 
- Simulaci můžeme nejprve spustit a poté pozastavit
 - ručně - tlačítko Pause 
 - podmínkou
 - zastavení v čase
 - *DEBUG > Pause Time ...* nastavit čas
 - zastavení v hodnotě
 - podmíněný breakpoint
 - *DEBUG > Add Breakpoint*
 - (pravý klik na signál *> Add conditional breakpoint*)
 - zvolit typ porovnávání (<, >, =, ...)
 - nastavit hodnotu prahu
 - zobrazí ze graficky 
 - kliknutím na značku lze vypnout nebo přenastavit
- Krokování vpřed – tlačítko 
- Krokování zpět
 - zapnutí ... *Step Back*  *> ☒ Enable stepping back*
 - nastavit počet kroků, kolik si má pamatovat
 - nastavit „rozteč kroků“, které se budou pamatovat
 - krokování – tlačítko 
- Nastavení kroku
 - roletka *Step Back > Configure ...*  *> Move back/forward by*
- Zobrazení hodnot při krokování
 - standardní bloky – Scope, Display, ...
 - okamžité hodnoty signálu (zajímavé právě při krokování)

- označit signál > *DEBUG* > *Output Values* > 
- (pravý klik na signál > *Show Value Label of Selected Port*)

⇒ Úloha

→ Nastavit zobrazení hodnot signálů a vykonat následné podúlohy

- breakpoint časový ... 2s
 - simulovat, krokovat tam i zpět
- breakpoint hodnotový ... dráha $x = 50$
 - simulovat, krokovat



Parametry simulace

- *MODELING* > *Model Settings*, ikona 

Solver

- Nastavení parametrů řešiče
- Princip numerické metody:



- výpočet aproximace má určitou přesnost
- čím kratší je krok, tím přesnější (ale pomalejší) => snaží se o co nejdelší krok v dané přesnosti
- **Metody s proměnným krokem (Variable-step)**
 - volí kroky v závislosti na přesnosti


- nastavení relativních tolerancí, případně absolutních tolerancí v parametrech modelu, dílčí nastavení tolerancí v integrátoru (pro daný integrátor má přednost)
- **auto (Automatic solver selection)**
 - MATLAB zvolí numerickou metodu sám
- **ode45 (Dormand-Prince)**
 - metoda Runge-Kutta (4,5)
 - výpočet metodou 5. řádu s odhadem chyby 4. řádu
 - **nejvýkonnější pro nejširší škálu systémů** => proto bývá často zvolena při automatické volbě řešiče
 - pokud je výpočet metodou ode45 pomalý, může se jednat o tzv. stiff systém => použít některý ze stiff řešičů (viz. níže)
- **ode23 (Bogacki-Shampine)**
 - metoda Runge-Kutta (2,3)
 - nižší řád než ode45
 - může být efektivnější než ode45 v případě hrubého nastavení tolerancí nebo simulace systémů s mírným stiff chováním
- **ode113 (Adams)**
 - Adams-Bashforth-Moulton PECE řešič
 - může být efektivnější než ode45 v případě striktního nastavení tolerancí nebo pro výpočetně intenzivní úlohy
- někdy nelze dopředu říci, která metoda bude lepší – je třeba to zkusit
- **metody pro stiff systémy:**
 - ode15s, ode23s, ode23t, ode23tb
 - stiff systém je systém s rozdílem časových konstant $> 1:1000$, obsahuje jak velmi pomalé, tak rychlé děje
 - standardní metody velmi drobí krok – neúnosně pomalé
 - stiff metody navrženy pro simulaci těchto systémů, umožňují mnohem radikálnější změny kroku, atd.


⇒ Úloha

→ Nastavit Ode23s > simulovat > méně kroků

- **Metody s pevným krokem (Fixed-step)**
 - nelze volit přesnost (délka kroku a přesnost jsou závislé veličiny)
 - metody typu Runge-Kutta 1. až 5. řád, 8. řád
 - např. 1. řád je Eulerova metoda
 - fixed-step metody jsou z důvodu aplikací v reálném čase (tam se nemůže vracet kvůli přesnosti a přepočítávat)

Data Import/Export

- **Save to workspace or file**
 - *Time* – simulační čas – ve výchozím stavu zaškrtnuto, proměnná **tout**
 - *States* – stavy integrátorů, ...
 - *Output* – signály z bloků **Out** v kořenové vrstvě modelu 
 - formát ukládání *States* a *Output*

- *Array, Structure, Structure with Time, Dataset*
- *Final states* – na konci simulace uloží poslední stav bloků (integrátory, ...)
 - *Save final operating point* – uloží strukturu s kompletním popisem koncového stavu simulace
- *Signal logging*
 - data ze signálů označených jako logované
- ☒ *Single simulation output*
 - data nejsou ukládána do samostatných proměnných, ale ve formě položek jediné proměnné typu *SimulationOutput*
 - přístup k položkám tečkovou notací, např. *Out.Time*
- **Load from workspace**
 - *Input* – signál jde do bloku **In** v kořenové vrstvě modelu 
 - *Initial state* – z tohoto stavu budou bloky se stavem začínat
 - lze předat i strukturu *SimState* – simulace pak bude začínat přesně tam, kde skončila, i včetně simulačního času
 - možnost rozdělit jeden simulační běh do několika částí
 - např. nejprve nechat model ustálit v pracovním bodě, uložit *SimState*, a od tohoto místa pak spouštět další simulace
- **Additional parameters**
 - *Limit data points to last* – uloží pouze n posledních vzorků
 - *Output options*
 - *Refine output*: výchozí
 - *Produce additional output*: spočítá v časech daných vektorem + jako normální metoda jak potřebuje
 - *Produce specified output only*: uloží jen v zadaných časech

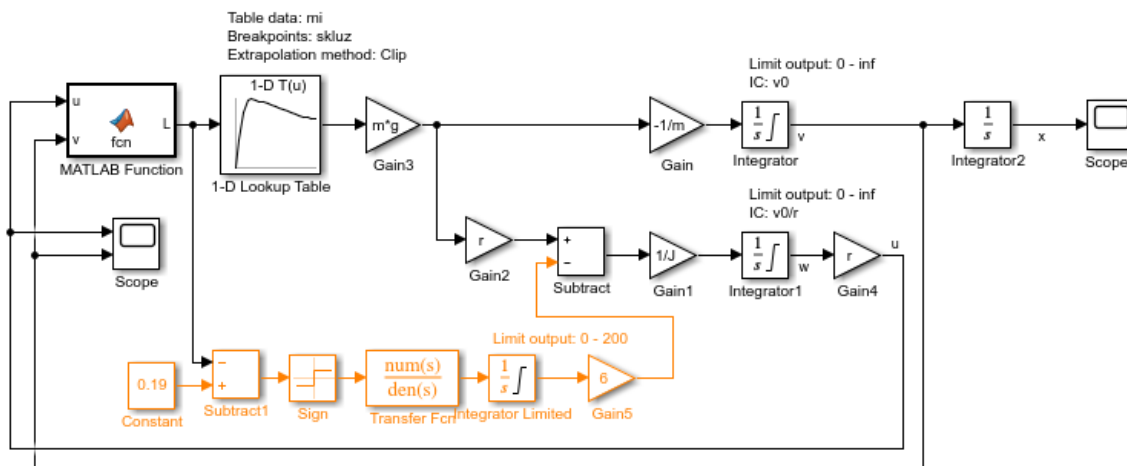
Pozn.: Při simulaci z příkazové řádky slouží bloky **Out** v kořenové vrstvě pro výstup simulační funkce a **In** pro vstupní hodnoty simulační funkce

Diagnostics

- Záložky – jak řeší potenciálně možné chybové stavy
 - nastavení 3 možností – neřeší, vypíše warning, způsobí error
- Uzel *Data Validity* – v *Advanced parameters* globální nastavení pro Model Verification
- V hlavním uzlu *Diagnostics* > 1. položka *Algebraic Loop* (školení Simulink 2)

Subsystémy a knihovny

Dokončení modelu: ABS

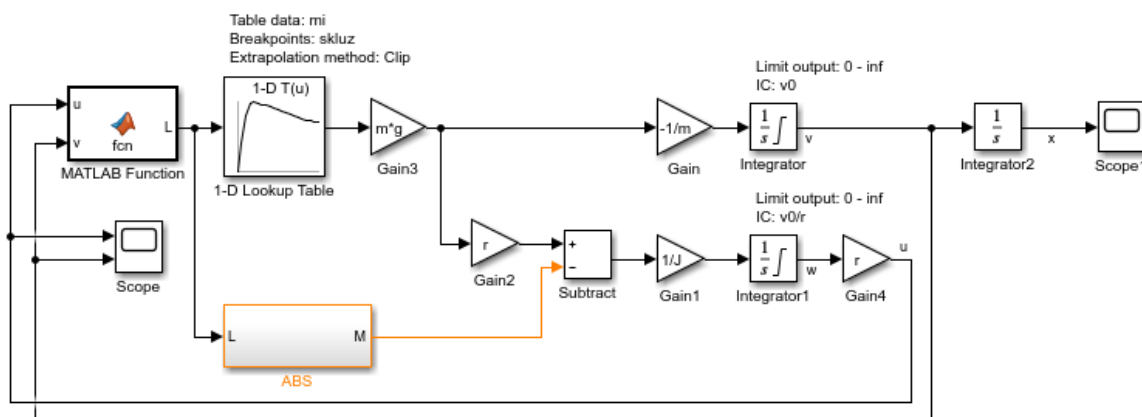






⇒ Úloha

- Otevřít model ABS v souboru `systemABS.slx`
- Nakopírovat do modelu brzdění

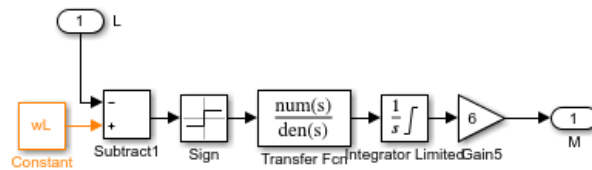
Subsystémy

- Rozčlenění modelu do logických jednotek
- Vytvoření:

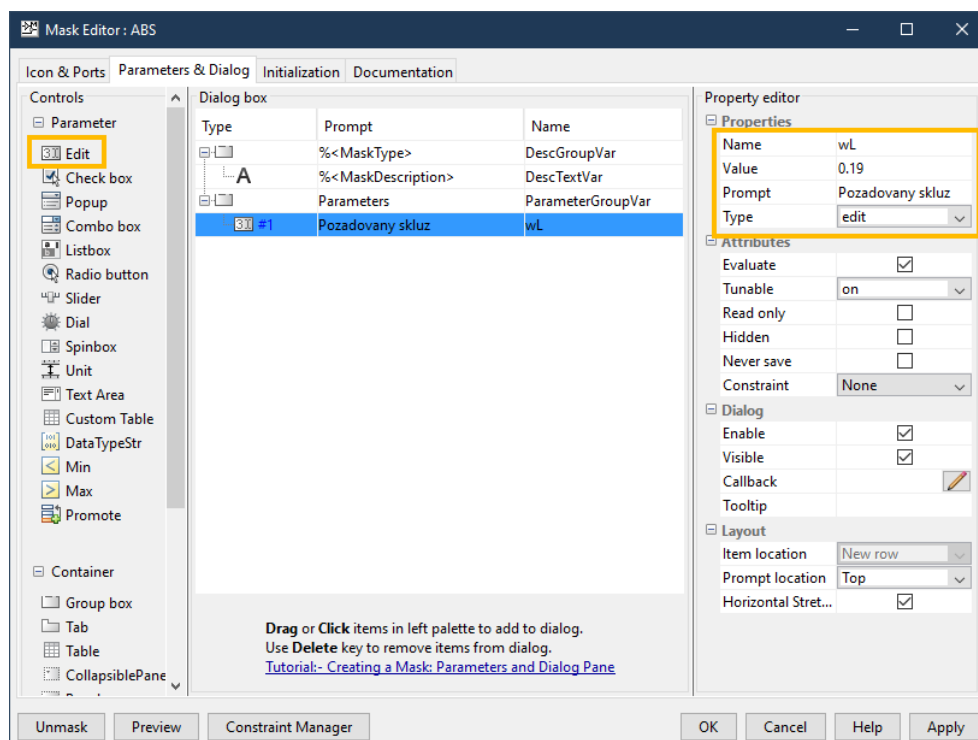


- označit bloky, které mají být součástí subsystému
 - **MULTIPLE** > *Create Subsystem*
 - (pravý klik > *Create Subsystem from Selection*)
 - lze do libovolné úrovně zanoření
- Navigace v hierarchii modelu
 - dvojklik ... otevření,  ... o patro výše
 - **Explorer Bar** -  ABS >  ABS control ... jako procházení složek ve Win7
 - **Model Browser** – stromová struktura modelu 
 - otevření subsystému v
 - novém panelu – pravý klik > *Open In New Tab*
 - novém okně – pravý klik > *Open In New Window*
- Další využití
 - uchovávat bloky v knihovnách a přetahovat hotové skupiny
 - aby se blok neotevřel, ale jen se objevila maska pro zadání parametrů

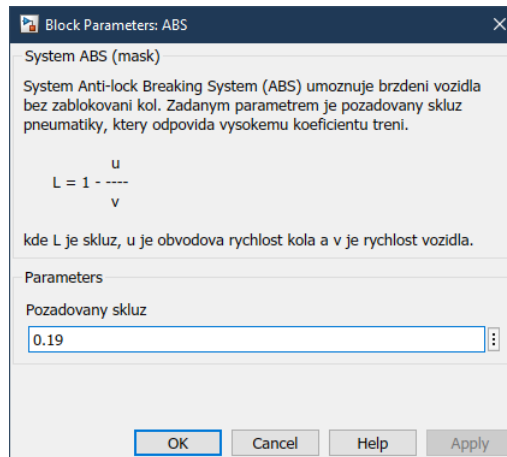
- Maskování subsystému
 - parametrizovat bloky v subsystému – do parametrů zadat proměnné, které budou předávány z masky



- označit subsystém > *SUBSYSTEM BLOCK* > *Create Mask*
- (pravý klik na subsystém > *Mask* > *Create Mask*)



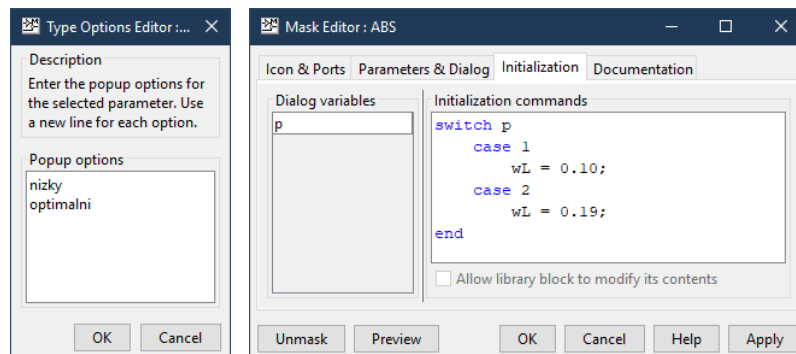
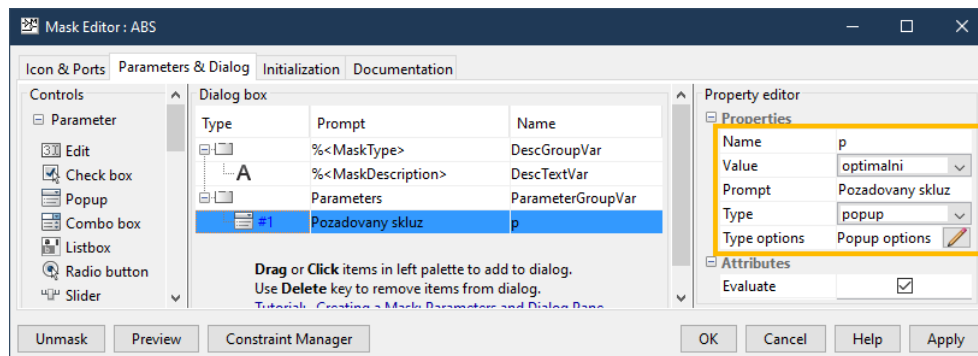
- záložka *Parameters & Dialog*
 - přidání prvku do masky
 - Controls* > kliknout na příslušnou ikonku
 - do políčka *Name* zadat jméno proměnné, které bude hodnota předána
 - do políčka *Prompt* zadat text popisku, který uživatel uvidí nad prvkem
 - typ ovládacích prvku lze měnit v políčku *Type*
 - edit, checkbox, výběr z výčtu - popup
 - Evaluate* – zda má být parametr vyhodnocen jako výraz v MATLABu nebo předán jako řetězec – parametr může být i funkce s proměnnými z workspace ('sin(a+b)')
 - Tunable* – zda má být parametr laditelný (měnitelný) za běhu simulace (např. Gain lze, Mux nelze)
- po dvojkliku na subsystém se objeví maska



- vnitřek subsystému
 - ikona ↓
 - (pravý klik > *Mask* > *Look Under Mask*)
- editace masky
 - označit subsystém > *SUBSYSTEM BLOCK* > *Edit Mask*
 - (pravý klik > *Mask* > *Edit Mask*)
- další záložky:
 - **Icon & Ports**
 - lze nastavit ikonu – jednoduché čárové kreslení, text, obrázek
 - dole je list s nápovědou syntaxe
 - **Documentation**
 - 1. řádek – „nadpis“ masky
 - 2. řádek – popis v masce, který je zobrazen nad řádky pro zadávání parametrů
 - 3. řádek – nápověda
 - zobrazí se po stisku tl. Help v masce
 - lze i rozsáhlý help
 - lze i odkaz na webovou stránku: web(‘ xxx ‘) – raději na lokální html soubor dodávaný s produktem
 - **Initialization**
 - příkazy, které se provedou po potvrzení masky – operují nad parametry zadávanými v masce a v modelu
 - pokud chci mít jinou sadu parametrů pro uživatele a jinou pro Simulink => mohu naprogramovat přepočít

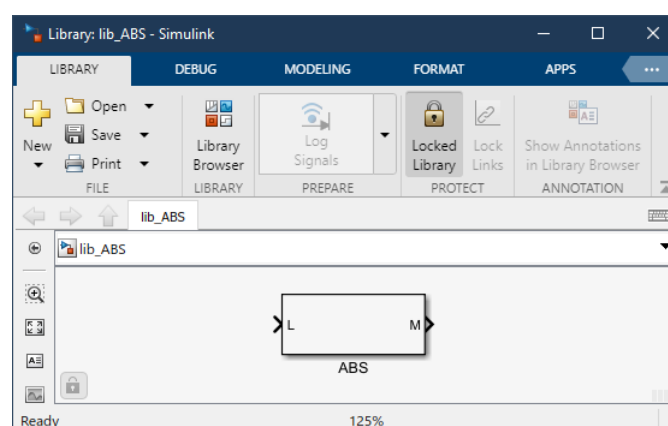
⇒ Úloha

→ Využití roletky:



Knihovny

- Když chci nějaký subsystém v novém modelu, najdu model, kde už je hotovo, a subsystém zkopíruji
 - nevýhoda: když chci pak něco změnit, musím udělat ve všech modelech
- Řeší knihovna – jediný fyzický výskyt, v modelech jsou pouze odkazy do knihovny
- Vytvoření knihovny
 - *SIMULATION* > *New* > *Library*
 - do knihovny vložím blok
 - knihovnu uložím



- Použití bloku z knihovny
 - otevřu knihovnu > přetáhnu blok do modelu
 - po dvojkliku v masce napsáno [mask][link] – je to jen odkaz do knihovny
- Nevýhody
 - s modelem musím distribuovat i knihovnu
 - pokud nechci – pravý klik > *Library Link* > *Disable Link*

- bloky z knihovny se nakopírují do modelu
- zůstane ještě informace, že blok pocházel z dané knihovny
- lze zvolit *Library Link > Resolve Link* – volba mezi restorováním bloku z knihovny do modelu a aktualizací knihovny
- pokud chci úplně odstranit vazbu na knihovnu *Library Link > Break Link*
- Knihovna je zamčená => pokud chci v ní něco upravit, musím potvrdit její odemčení (*LIBRARY > Locked Library*)

Využívání vlastní knihovny

- Pouze v pracovní složce
 - knihovna je vidět, pokud jí máme v pracovní složce
 - využití knihovny pouze s modely ve stejné složce
- Využívání vlastní knihovny stejně, jako vestavěných knihoven
 - složka s knihovnou musí být v cestách MATLABu
 - knihovnu otevírá zadáním jména: >> *mojelib*
 - při otevření a simulaci modelu využívajícího tuto knihovnu si ji MATLAB v cestách najde sám
- Jak vložit složku s knihovnou na cestu?
 - Napevno uložit do seznamu cest
 - MATLAB: *Home > Set Path*
 - grafické okno > přidám složku do seznamu tl. *Add Folder > Save*
 - Cesta se uloží k ostatním cestám v souboru *pathdef.m*
 - Nevýhoda
 - soubor je u instalace => musím mít právo zápisu (lze uložit i do spouštěcí složky MATLABu)
 - ! soubor svázan s danou verzí MATLABu => s novou verzí MATLABu musím udělat znovu
 - Dynamické přidání cest
 - příkaz: *addpath('c:/matlab/myfiles')*
 - nevýhoda
 - jen po dobu trvání spuštění MATLABu
 - řešení:
 - příkazy *addpath* uložit do souboru *startup.m*
 - *startup.m* umístit do spouštěcí složky MATLABu
 - *startup.m* se automaticky spouští při každém startu MATLABu
 - => trvalé, nezávislé na verzi MATLABu

Přidání knihovny do Library Browseru

- Postup uveden v dokumentaci
 - >> doc "Add Libraries to the Library Browser"

Tipy pro modelování

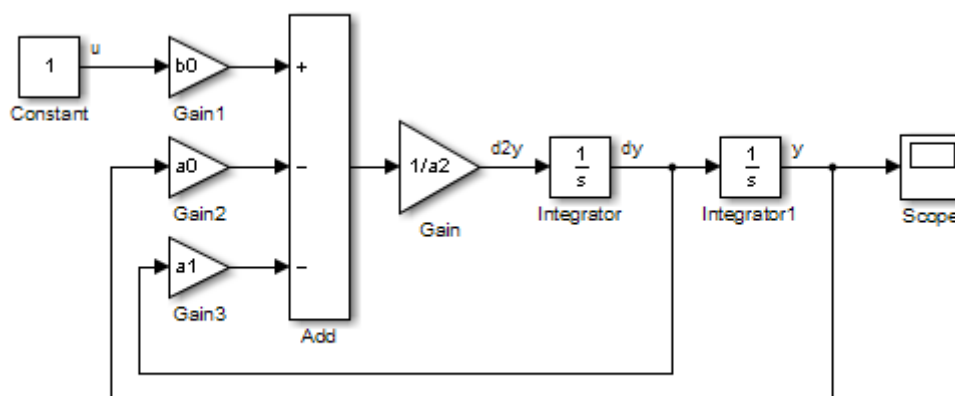
Modelování rovnic vyšších řádů a soustavy rovnic

- Soustava vyššího řádu
 - vyjádření nejvyšší derivace
 - n-tá derivace => série n integrátorů
 - vstup do prvního integrátoru dán rovnicí $x^n = \dots$
- Soustava více rovnic
 - v každé rovnici vyjádřím proměnnou s nejvyšší derivací (každou proměnnou ale jen jednou)
 - pro každou proměnnou i v n_i -té derivaci n_i integrátorů
 - vstup do prvního integrátoru dán danou rovnicí
 - rovnice obsahuje i vstupy z ostatních rovnic

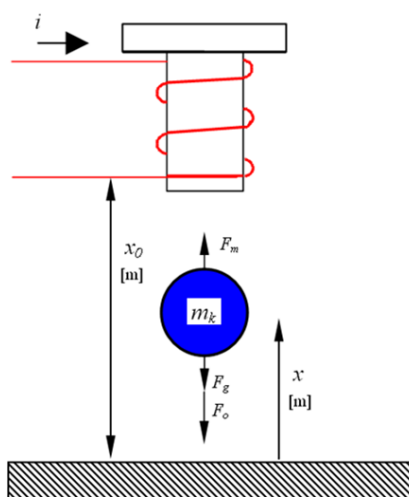
Příklad: Lineární rovnice 2. řádu:

$$a_2 \ddot{y} + a_1 \dot{y} + a_0 y = b_0 u \quad \Rightarrow \text{upravíme na:}$$

$$\ddot{y} = \frac{1}{a_2} (b_0 u - a_0 y - a_1 \dot{y})$$



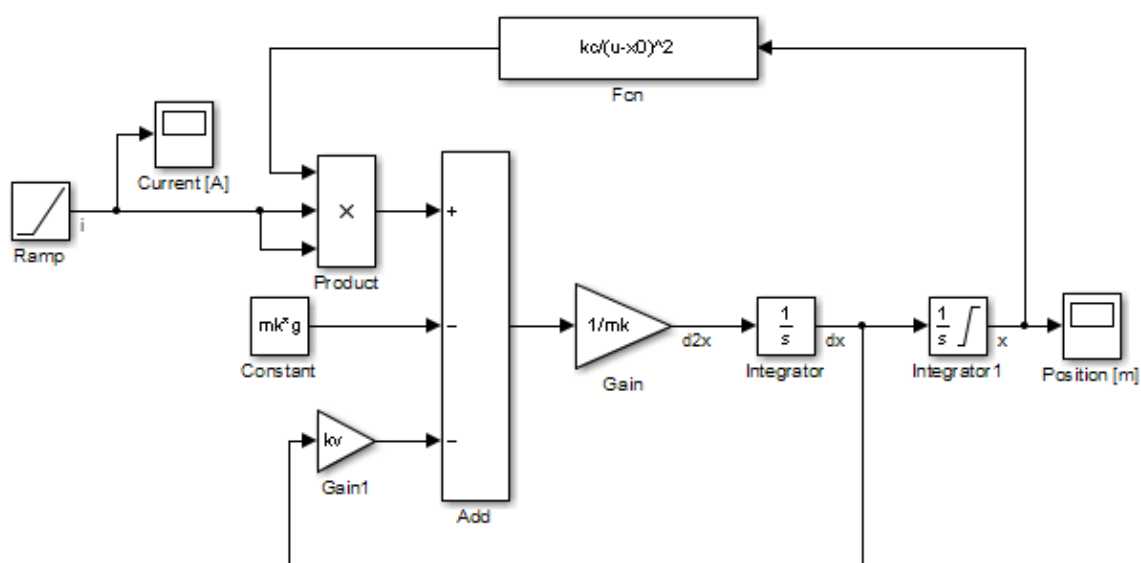
Příklad: Nelineární soustava 2. řádu – Magnetická levitace



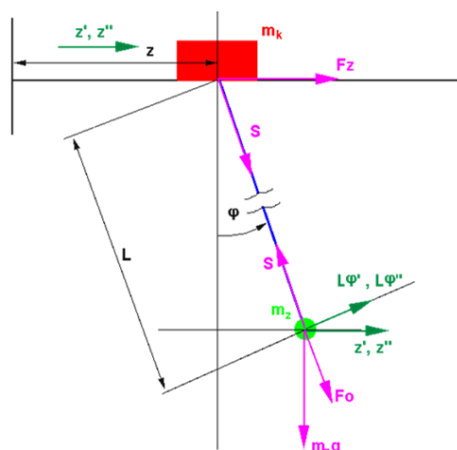
$$m_k \ddot{x} = F_m - F_g - F_o$$

$$m_k \ddot{x} = k_c \frac{i^2}{(x_0 - x)^2} - m_k g - k_v \dot{x}$$

$$\ddot{x} = \frac{1}{m_k} \left(k_c \frac{i^2}{(x_0 - x)^2} - m_k g - k_v \dot{x} \right)$$



Příklad: Soustava nelineárních rovnic 2. řádu – Jeřábová kočka



$$m_k \ddot{z} = F_z + S \sin \varphi$$

$$m_z (L \ddot{\varphi} + \ddot{z} \cos \varphi) = -m_z g \sin \varphi$$

$$m_z \ddot{z} \sin \varphi = F_o + m_z g \cos \varphi - S$$

$$F_o = m_z L \dot{\varphi}^2$$

$$S = -m_z \ddot{z} \sin \varphi + m_z L \dot{\varphi}^2 + m_z g \cos \varphi$$

$$(m_k + m_z \sin^2 \varphi) \ddot{z} = F_z + (m_z L \dot{\varphi}^2 + m_z g \cos \varphi) \sin \varphi$$

$$m_z L \ddot{\varphi} = -m_z g \sin \varphi - m_z \ddot{z} \cos \varphi$$

$$\ddot{z} = \frac{F_z + (m_z L \dot{\varphi}^2 + m_z g \cos \varphi) \sin \varphi}{m_k + m_z \sin^2 \varphi}$$

$$\ddot{\varphi} = \frac{1}{m_z L} (-m_z g \sin \varphi - m_z \ddot{z} \cos \varphi)$$

- \ddot{z} je výstupem první rovnice, proto se ve druhé rovnici zavede pouze jako vstup

