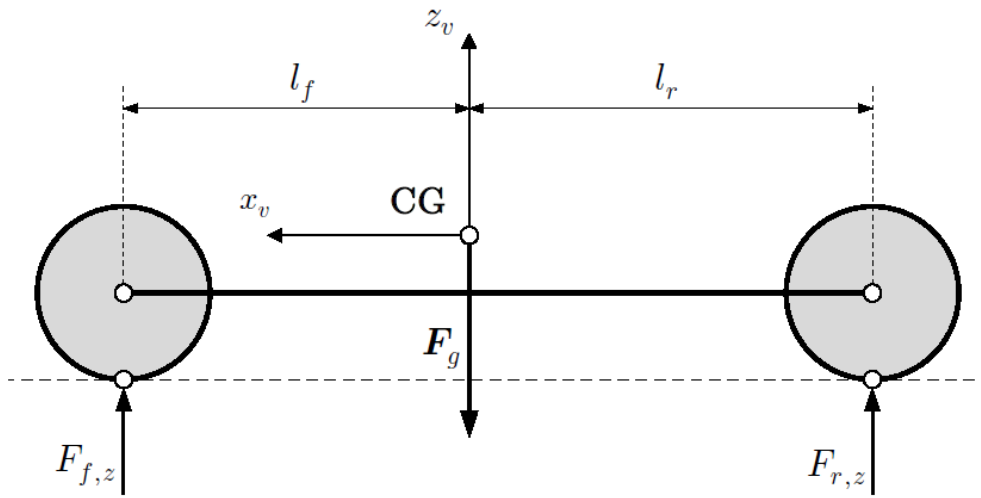
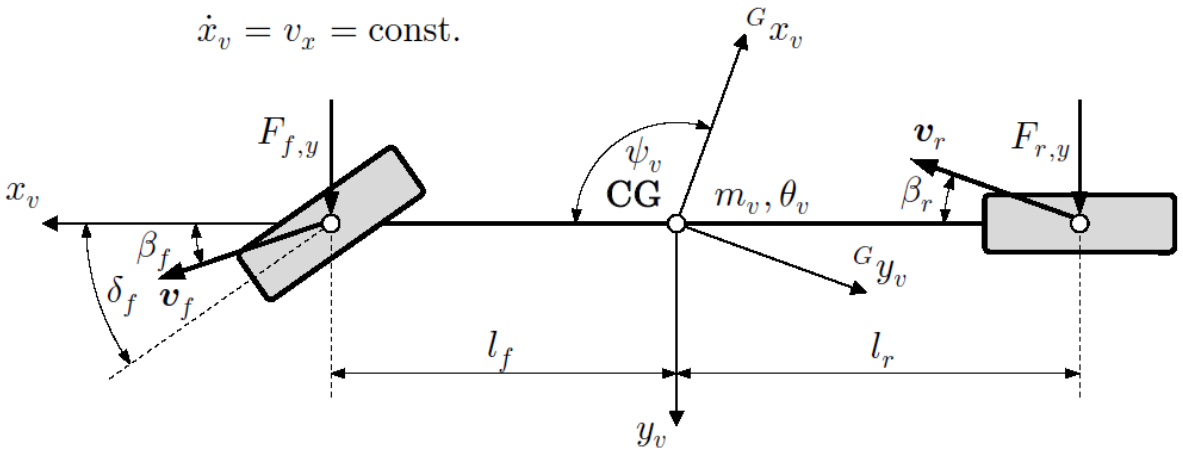
MATLAB OKTATÁS
6. ELŐADÁS
SIMULINK ALAPOK
JÁRMŰMÉRNÖKI ALKALMAZÁSSAL

Dr. Bécsi Tamás
Hegedüs Ferenc

FIZIKAI MODELL

- Egy nyomvonalú lineáris járműmodell (biciklimodell).
  - Az első és hátsó tengelyeken lévő kerékpárokat egy-egy virtuális kerékkel reprezentálja.
  - A jármű tömegközéppontjának helye időben állandó.
  - A jármű hosszirányú sebessége állandó.
  - A járműváz síkbeli merev test, három szabadságfokkal:
    - x hosszirányú (x-irányú) elmozdulás
    - y oldalirányú (y-irányú) elmozdulás
    - ψ függőleges tengely körüli (z-irányú) elfordulás
  - A modell paraméterei:
    - v hosszirányú sebesség
    - m tömeg
    - J tehetetlenségi nyomaték a z-tengely körül
    - l_f, l_r az első és hátsó tengely és a tömegközéppont vízszintes távolsága
    - c_f, c_r az első és hátsó tengelyek kanyarmerevsége
  - A modell bemenete:
    - δ kormányszög az első keréken
  - A modell kimenete:
    - $\dot{\psi}$ legyezési szögsebesség





MATEMATIKAI MODELL: MOZGÁSEGYENLETEK

- Oldalirányú gyorsulás

$$\ddot{y}(t) = -\frac{g(c_f l_r + c_r l_f)}{v(l_f + l_r)}\dot{y}(t) - \left(v + \frac{g l_f l_r (c_f - c_r)}{v(l_f + l_r)}\right)\dot{\psi}(t) + \frac{g c_f l_r}{l_f + l_r}\delta(t)$$

$$\ddot{y}(t) = a_{11}\dot{y}(t) + a_{13}\dot{\psi}(t) + b_{11}\delta(t)$$

- Függőleges tengely körüli szöggyorsulás

$$\ddot{\psi}(t) = -\frac{m g l_f l_r (c_f - c_r)}{J v(l_f + l_r)}\dot{y}(t) - \frac{m g l_f l_r (c_f l_f + c_r l_r)}{J v(l_f + l_r)}\dot{\psi}(t) + \frac{m g c_f l_f l_r}{J(l_f + l_r)}\delta(t)$$

$$\ddot{\psi}(t) = a_{31}\dot{y}(t) + a_{33}\dot{\psi}(t) + b_{31}\delta(t)$$

- A jármű sebessége a földhöz viszonyítva

$$\dot{x}_G = \dot{x}\cos(\psi) - \dot{y}\sin(\psi), \qquad \dot{y}_G = \dot{x}\sin(\psi) + \dot{y}\cos(\psi)$$

MATEMATIKAI MODELL: ÁLLAPOTTÉR REPREZENTÁCIÓ

- A mozgásegyenlet mátrixos alakban

$$\begin{bmatrix} \ddot{y} \\ \dot{y} \\ \ddot{\psi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 \\ 1 & 0 & 0 & 0 \\ a_{31} & 0 & a_{33} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{y} \\ y \\ \dot{\psi} \\ \psi \end{bmatrix} + \begin{bmatrix} b_{11} \\ 0 \\ b_{31} \\ 0 \end{bmatrix} \delta, \qquad Y = \dot{\psi}$$

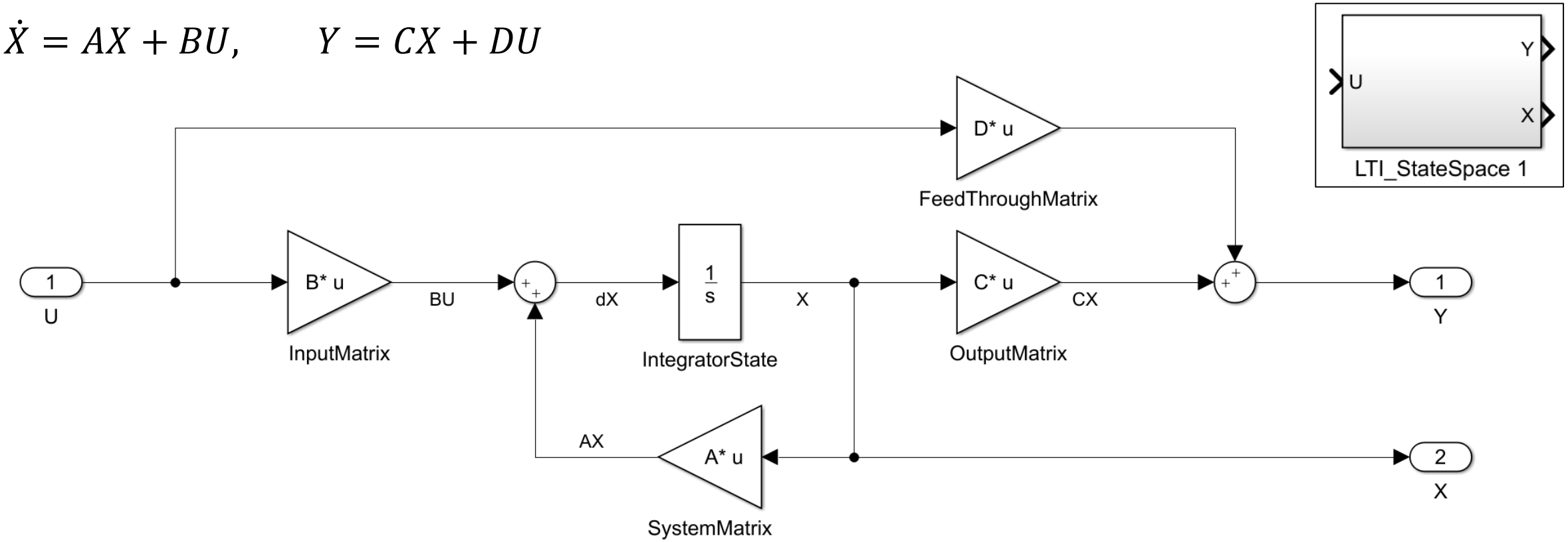
- Lineáris időinvariáns állapottér reprezentáció

$$\dot{X} = AX + BU, \qquad Y = CX + DU$$

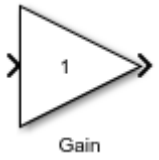
$$X = \begin{bmatrix} \dot{y} \\ y \\ \dot{\psi} \\ \psi \end{bmatrix}, \qquad U = \delta, \qquad A = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 \\ 1 & 0 & 0 & 0 \\ a_{31} & 0 & a_{33} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} \\ 0 \\ b_{31} \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \qquad D = 0$$

SIMULINK MODELL – ÁLLAPOTTÉR ALRENDSZER

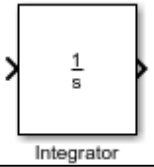
$$\dot{X} = AX + BU, \qquad Y = CX + DU$$



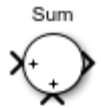
- *Gain* (erősítés)
  - konstanssal való szorzás



- *Integrator* (integrátor)
  - bemenő jel integrálása

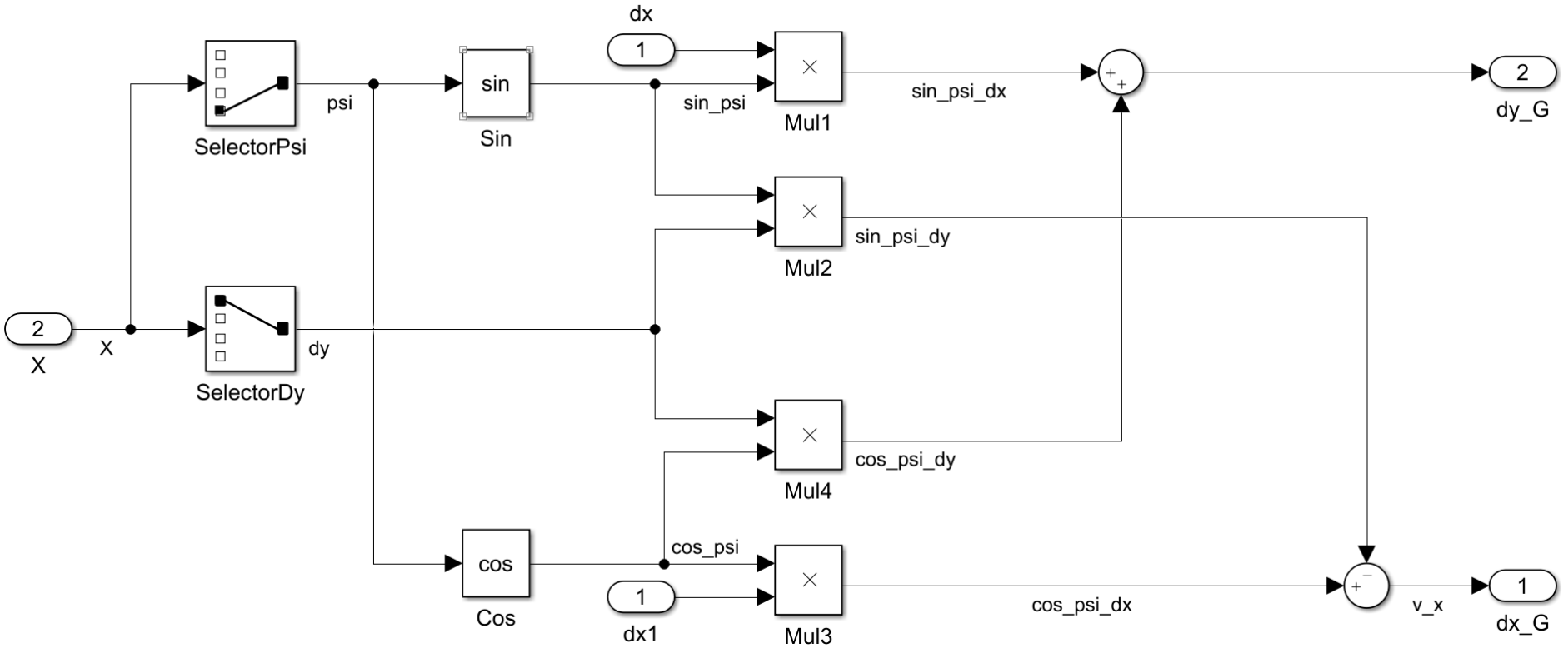


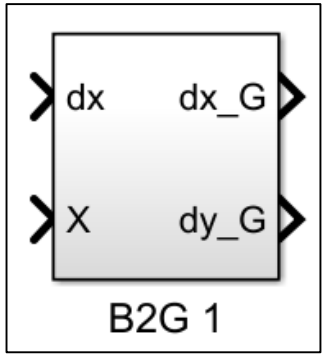
- *Sum* (összegző)
  - bemenő jelek összeadása, kivonása



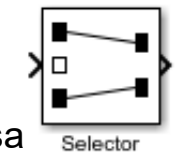
SIMULINK MODELL – ABSZOLÚT SEBESSÉG ALRENDSZER

$$\dot{x}_G = \dot{x}\cos(\psi) - \dot{y}\sin(\psi), \qquad \dot{y}_G = \dot{x}\sin(\psi) + \dot{y}\cos(\psi)$$

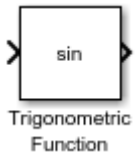




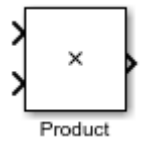
- *Selector* (választó)
  - vektor jelből adott komponensek kiválasztása



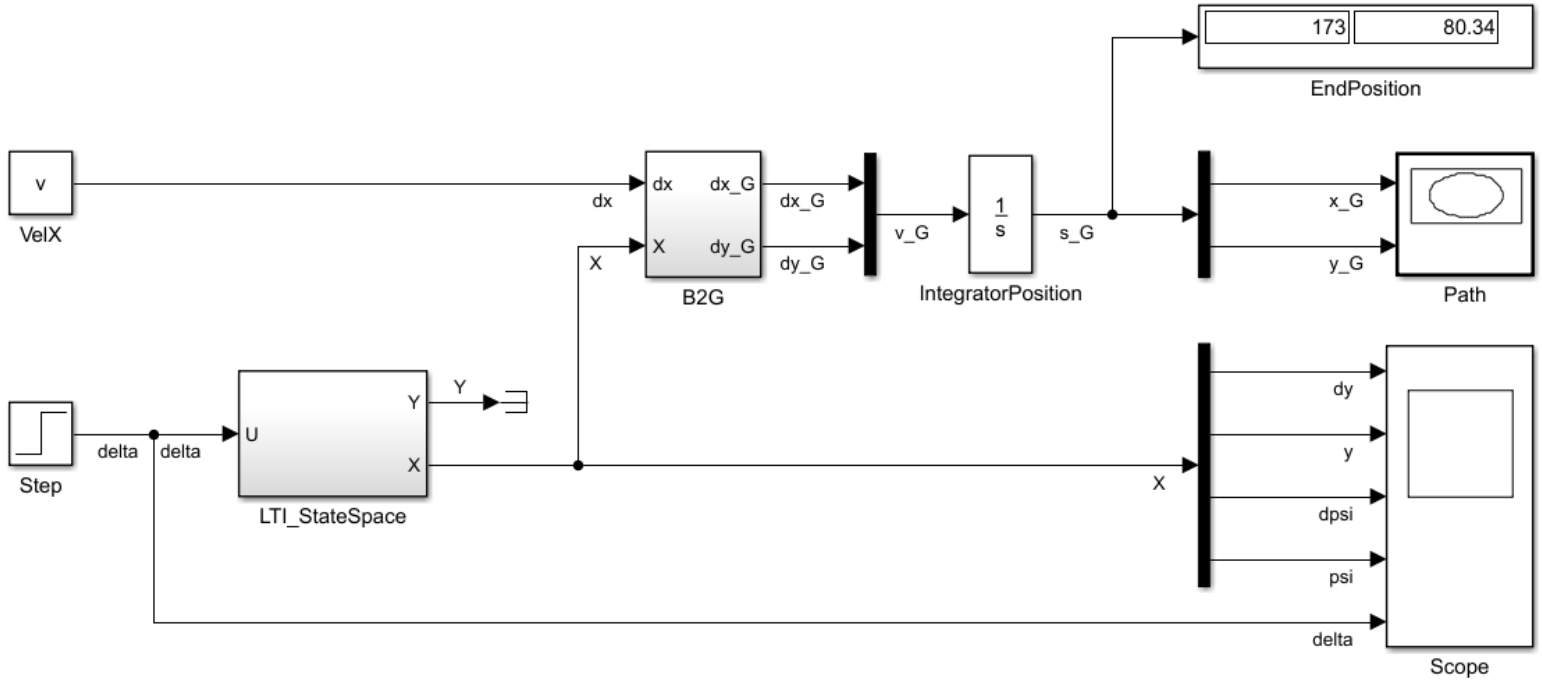
- *Trigonometric Function* (trigonometrikus fv.)
  - sin, cos, tan, ...



- *Product* (szorzó)
  - bemenő jelek szorzása, osztása

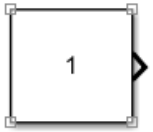


SIMULINK MODELL – TELJES RENDSZER

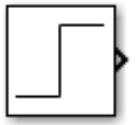


- *Demux* (demultiplexer)
  - vektorjelek komponensekre bontása (demultiplexálás)

- *Constant* (állandó)
  - konstans érték

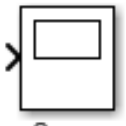


- *Step* (ugrás)
  - ugrás-függvény



- *Mux* (multiplexer)
  - jelek vektorba foglalása (multiplexálás)

- *Scope* (megjelenítő)
  - bemenő jelek megjelenítése



- *XY Graph* (megjelenítő)
  - bemenő jelek alapján X-Y ábra felrajzolása

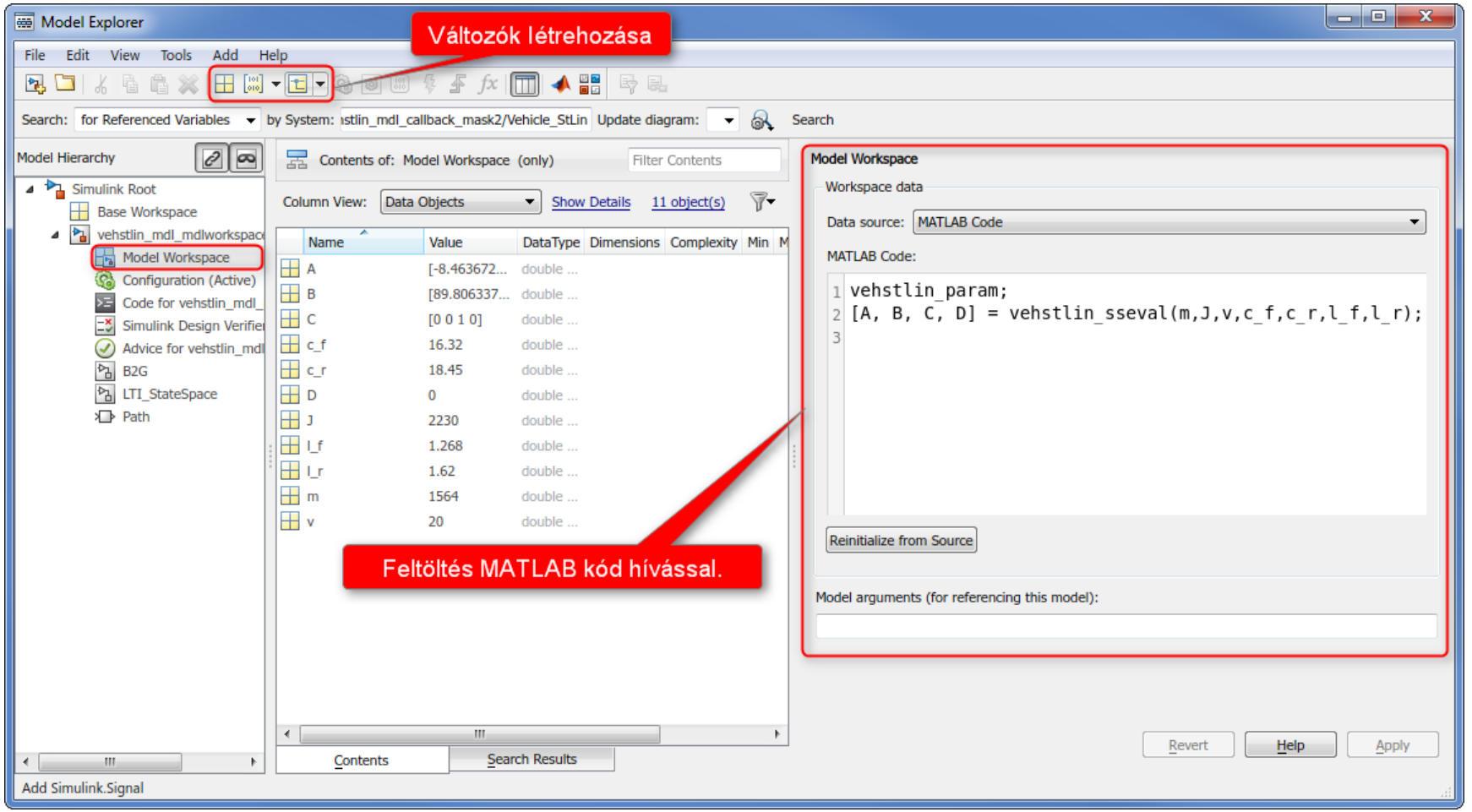
- *Display* (kijelző)
  - bemenő jel aktuális értékének felrajzolása

MODELL PARAMÉTEREK

- A Simulink modell paraméterezése manuálisan a blokkokra való dupla kattintás után a paraméter párbeszédablakban történik. A paraméter lehet:
  - Literális konstans érték.
  - MATLAB alapértelmezett munkaterületén (*Base Workspace*) létező változó.
    - MATLAB ablakban a Workspace lapon látható változók, amelyeket a MATLAB *Command Window*-ba írt parancsokkal, illetve a MATLAB script-ek futtatásával hozunk létre.
  - Simulink modell munkaterületén (*Model Workspace*) létező változó.
    - Simulink ablakban *View* > *Model Explorer* > *Model Explorer* vagy Ctrl+H segítségével előhívható *Model Explorer*-ben tekinthető meg a *Model Workspace*.
    - *Model Workspace* változó létrehozható manuálisan a *Model Explorer*-ben.
    - *Model Workspace* változó létrehozható a MATLAB-ból programozva is.
    - A teljes *Model Workspace* a modell megnyitásakor inicializálható MAT-fájlból, vagy tetszőleges MATLAB-kód futtatásával.

MODEL WORKSPACE PÉLDA

- A jármű paramétereinek automatikus betöltése, és azok alapján az állapotegyenlet mátrixainak automatikus kiszámítása a modell betöltésekor. Minden változó a *Model Workspace*-en fog tárolódni.

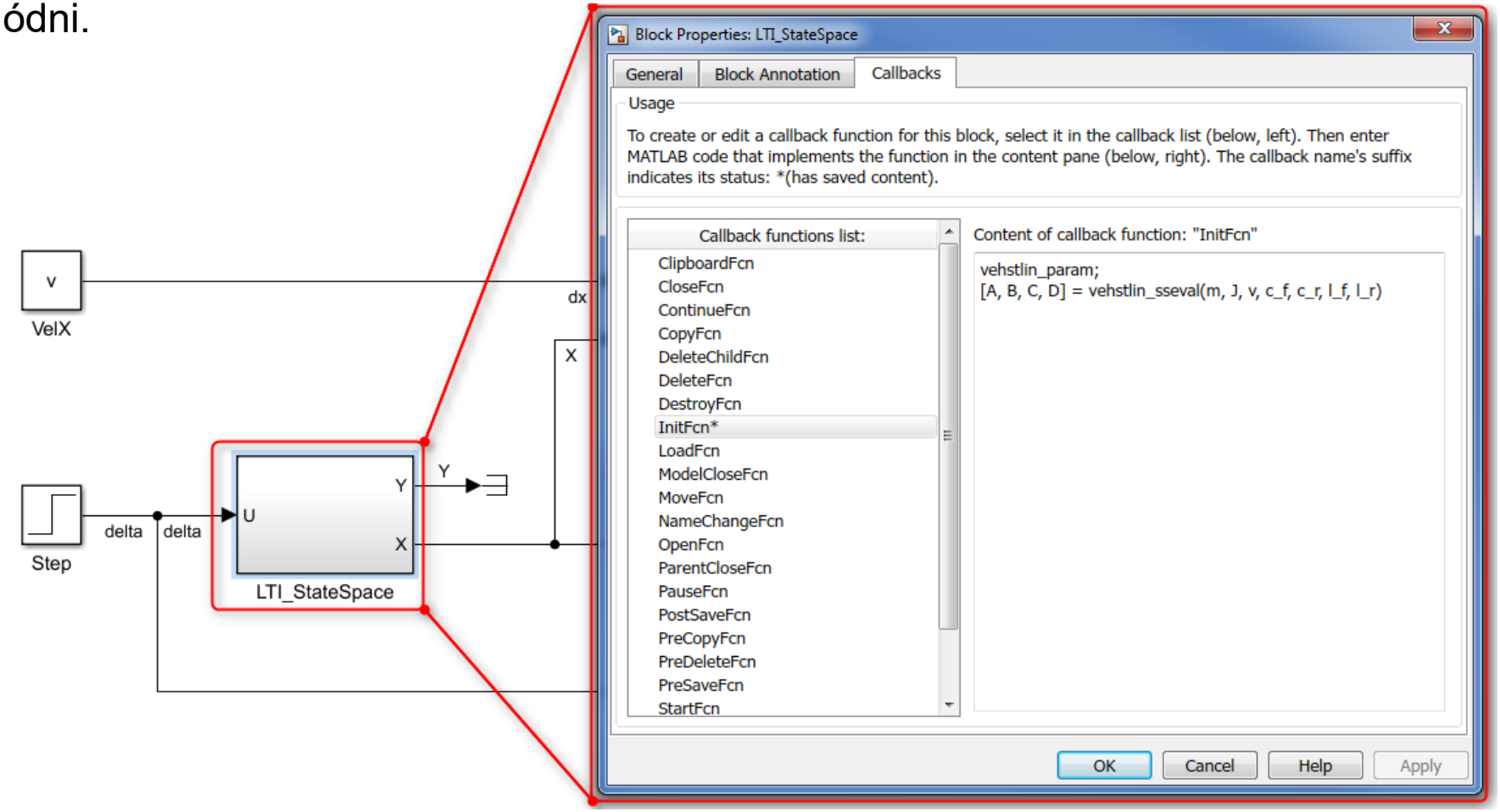


FÜGGVÉNY VISSZAHÍVÁS (FUNCTION CALLBACK)

- A Simulink modell, és benne minden egyes blokk is képes a szimuláció bizonyos meghatározott állapotaiban (modell betöltés előtt és után, inicializáláskor, futtatás után, stb.) MATLAB utasításokat végrehajtani, scripteket, függvényeket futtatni. Ezt a funkciót hívjuk függvény visszahívásnak (function callback).
- A teljes modell callback függvényeit a *File* > *Model Properties* > *Model Properties* > *Callbacks* fülön érjük el, míg az egyes blokkokét az adott blokkon jobb egérgombbal klikkelve a *Properties* > *Callbacks* fülön találjuk.
- Tipikus alkalmazások:
  - Paraméterek automatikus beolvasása, kiszámítása a futtatás előtt.
    - Független és szimulációnként állandó paraméterek betöltéséhez tipikusan a `PreLoadFcn` vagy `PostLoadFcn` (modell betöltése előtt illetve után futnak) használható.
    - Független paraméterek alapján számolódó egyéb paraméterek, szimulációnként változó értékek betöltéséhez az `InitFcn` vagy `StartFcn` (szimuláció előtt futnak) használható.
  - Szimulációs eredmények automatikus feldolgozása, kiértékelése a futtatás után.
    - Szimulációs eredmények kiértékeléséhez a `StopFcn` (szimuláció után fut) használható.

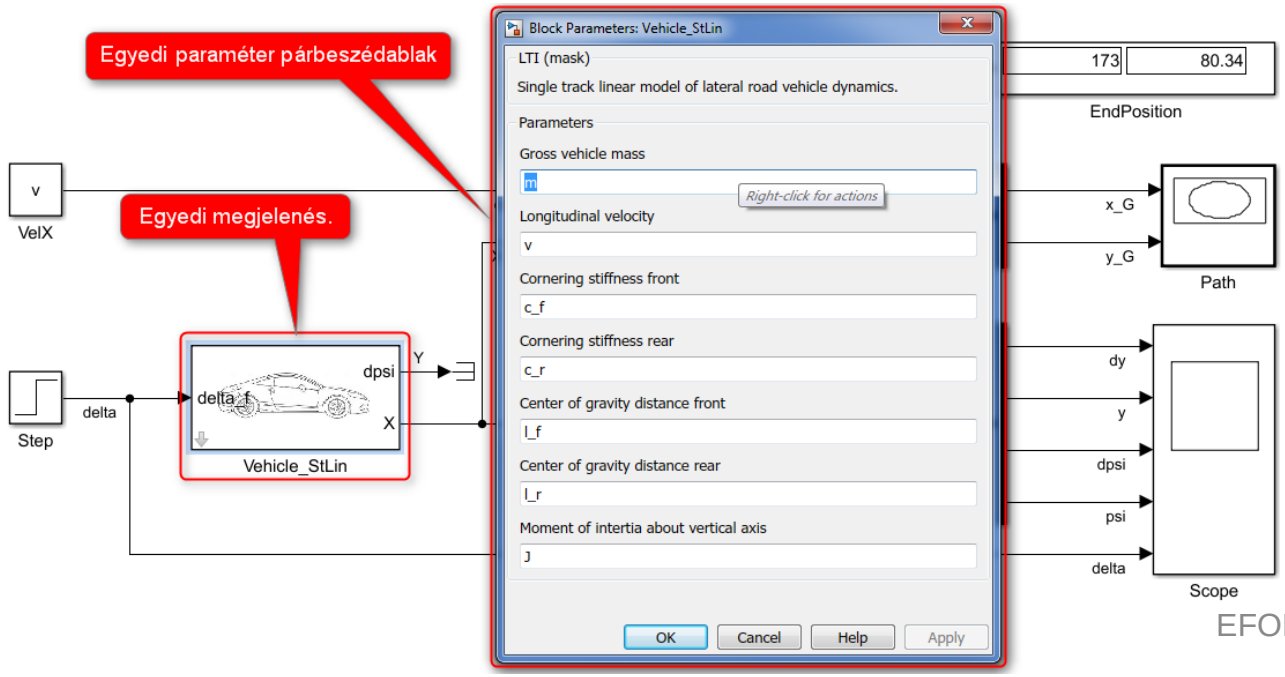
FÜGGVÉNY VISSZAHÍVÁS (FUNCTION CALLBACK) PÉLDA

- A jármű paramétereinek automatikus betöltése, és azok alapján az állapotegyenlet mátrixainak automatikus kiszámítása a szimuláció indításakor (`InitFcn`). Minden változó a Base Workspace-en fog tárolódni.



ALRENDSZER MASZK (SUBSYSTEM MASK)

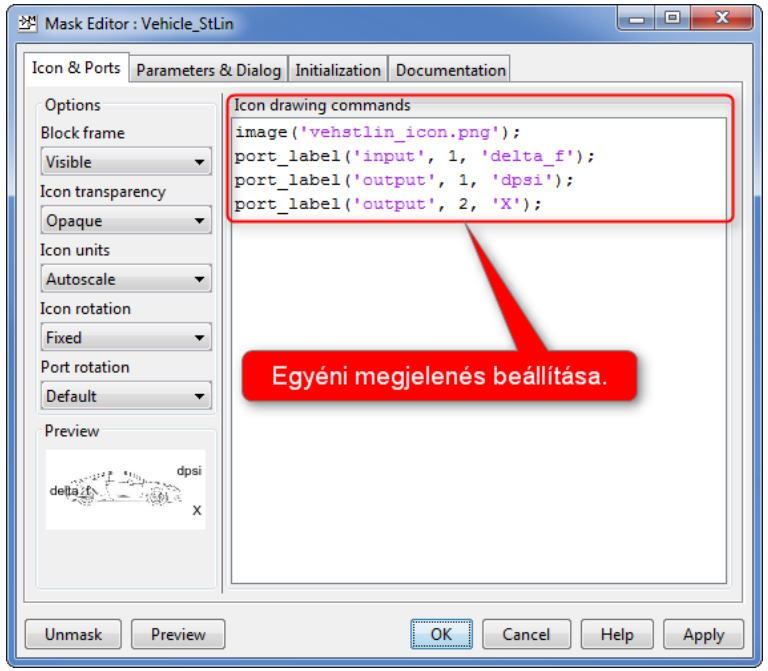
- Az egyes alrendszerek blokkjait ún. maszkkal láthatjuk el.
  Az alrendszer maszkolásával:
  - Egyéni felhasználói felületet tudunk megvalósítani: a maszkolt alrendszer blokkjának saját kinézete, paraméter párbeszédablaka, leírása, segítség szövege lesz.
  - További egyéni üzleti logikát (pl. paraméterek alapján további értékek kiszámítása) tudunk megvalósítani.
  - Elrejthetünk az adott alrendszerhez tartozó adatokat.

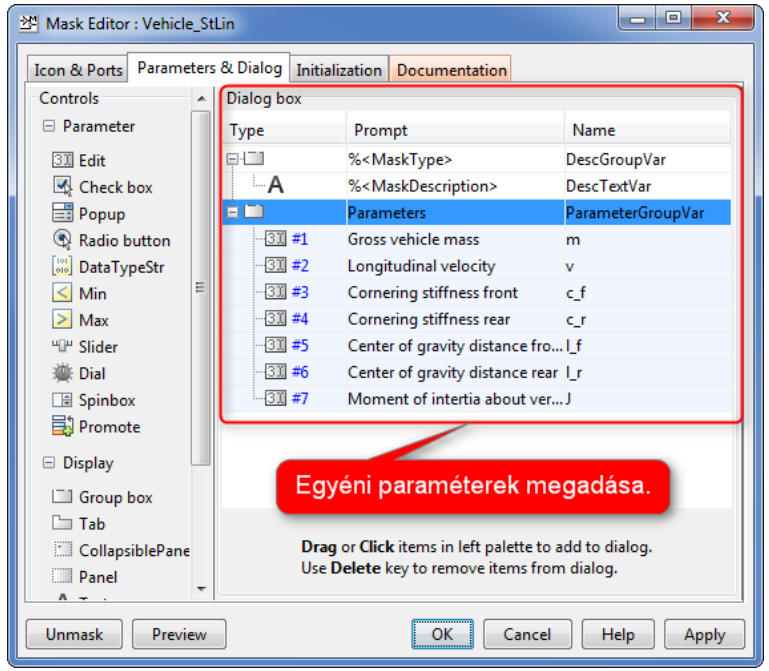


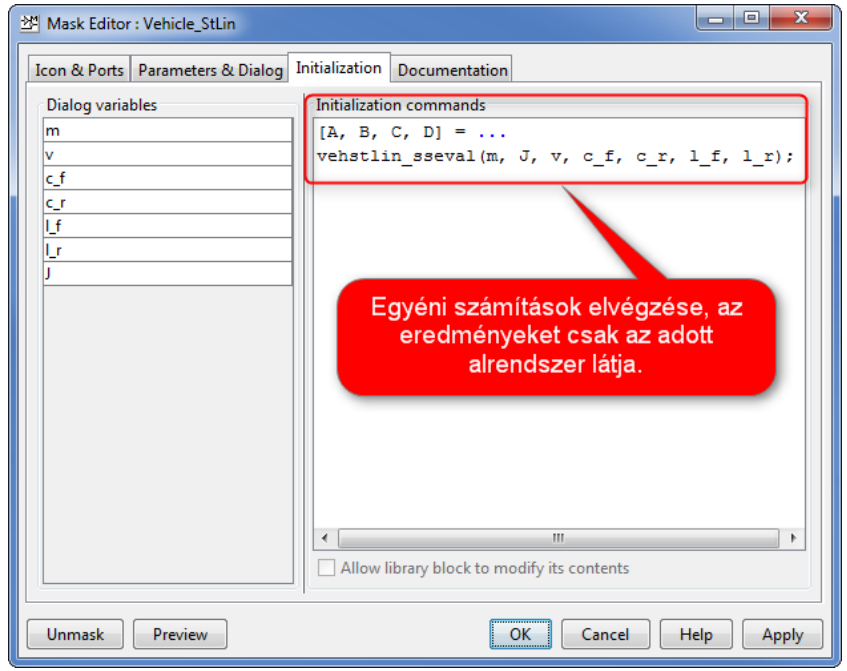
ALRENDSZER MASZK (SUBSYSTEM MASK) PÉLDA

sl/vehslin_mdl_callback_mask2.mdl
sl/vehstlin_icon.png

- Egyéni kinézet (kép betöltése, port feliratok) megadása (bal oldalon)
- Egyéni paraméter párbeszédablak és rendszerparaméterek megadása (középen).
- Az állapotegyenlet mátrixainak automatikus kiszámítása a maszkon megadott paraméterek alapján. A kiszámolt értékek rejtve vannak, csak a maszkolt alrendszer látja őket (jobb oldalon).







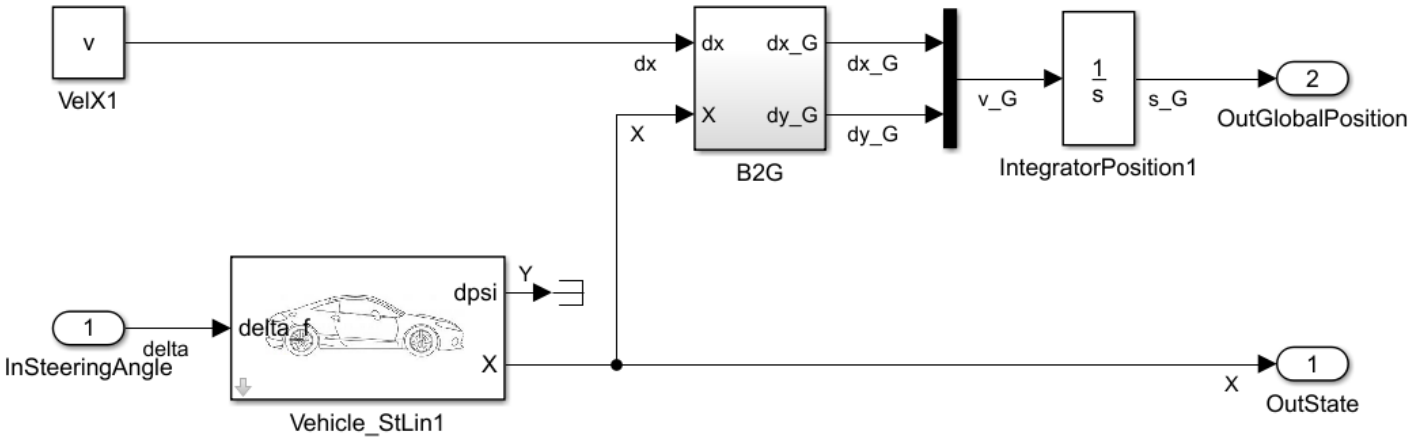
MATLAB BE- ÉS KIMENET

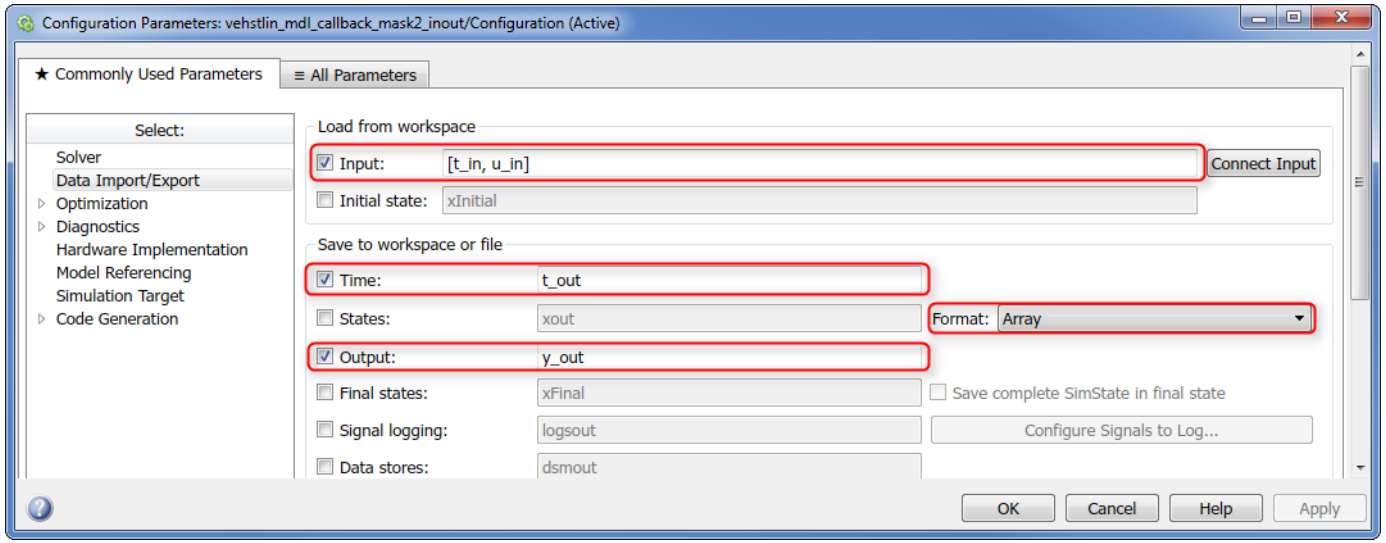
- A Simulink jelek MATLAB *Base Workspace*-re való exportálása, valamint onnan történő importálása többféleképpen is megoldható.
  - Modell szintű be- és kimeneti portok alkalmazásával.
    - A működés a *Simulation* > *Model Configuration Parameters* > *Data Import / Export* menüpont alatt állítható be.
  - *FromWorkspace* és *ToWorkspace* blokkok segítségével.
    - A működés az egyes blokkok paraméter párbeszédablakában állítható be.
- A bemeneti és kimeneti jelek többféle formátumban megadhatóak illetve megkaphatóak, ezek közül a leggyakrabban használtak, és a portok valamint a *From/ToWorkspace* blokkok esetén egyaránt használhatóak:
  - Tömb formátum (*Array*).
    - Bemenet: mátrix, melynek első oszlopában időadatok, a többi oszlopában pedig a bemenet adott időpillanatokhoz tartozó értékei vannak.
    - Szimulált időpillanatok: oszlopvektor, mely a szimulált időpillanatokat tartalmazza.
    - Kimenet: mátrix, melynek minden sora az adott szimulált időpillanathoz tartozó kimenetvektort tartalmazza.
  - Időadattal ellátott struktúra (*Structure with time*).
  - MATLAB idősor objektum (*MATLAB Timeseries*).

MATLAB BE- ÉS KIMENET PÉLDA

sl/vehslin_mdl_callback_mask2_inout.mdl
sl/vehstlin_icon.png
sl/vehstlin_input.m

- Jel importálás / exportálás modell szintű portokkal.






```
% vehstlin_input.m
%% Time of simulation
t_start = 0;
t_step  = 0.1;
t_stop  = 14;
%% Input data of simulation
% Time data
t_in = (t_start:t_step:t_stop)';
% Steering angle excitation data
u_in = (1 / 180 * pi) *...
       ((t_in > 2.0) - ...
         2 * (t_in > 7.0) + ...
        (t_in > 12.0));
```

EFOP-3.6.3-VEKOP-16-2017-00001

AUTOMATIZÁLÁS 1

- Simulink modell megnyitása: **open_system** függvény

  open_system(mdlName)

  mdlName: a modell neve

- Simulink modell bezárása: **close_system** függvény

  close_system(mdlName)

  mdlName: a modell neve

- Simulink modell szimulációja: **sim** függvény

  sim(mdlName)

  mdlName: a modell neve

- Simulink modell vagy blokk paraméter értékének lekérdezése: **get_param** függvény

  parameterValue = get_param(objectName, parameterName)

  objectName: a modell neve vagy a blokk teljes elérési útja a modellen belül
  parameterName: a lekérdezni kívánt paraméter neve
  parameterValue: a lekérdezett paraméter értéke

- Simulink modell vagy blokk paraméter értékének beállítása: **set_param** függvény

  set_param(objectName, parameterName, parameterValue)

  objectName: a modell neve vagy a blokk teljes elérési útja a modellen belül
  parameterName: a beállítani kívánt paraméter neve
  parameterValue: a beállítani kívánt paraméterérték

AUTOMATIZÁLÁS 2

- Simulink aktuális aktív modell neve: **`bdroot`** függvény

```
mdlName = bdroot
```

mdlName: az aktuális aktív modell neve

- Simulink aktuális aktív blokk elérési útvonala: **`gcb`** függvény

```
blkPath = gcb
```

blkPath: az aktuális aktív blokk elérési útvonala

- Adott nevű Simulink modell be van-e töltve: **`bdIsLoaded`** függvény

```
isLoaded = bdIsLoaded(mdlName)
```

mdlName: a modell neve

isLoaded: a modell be van-e töltve a memóriába, vagy sem (true/false)

AUTOMATIZÁLÁS PÉLDA

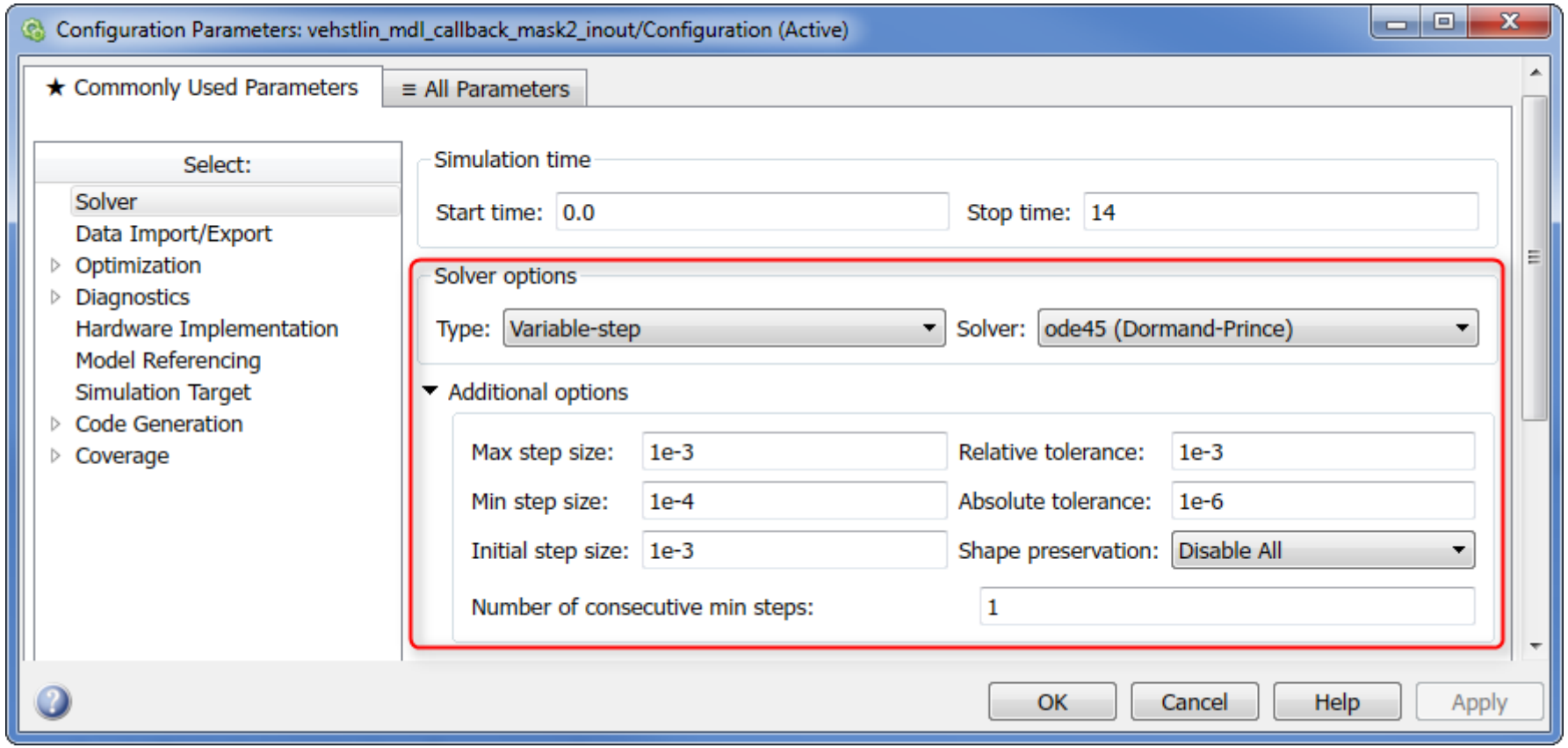
sl/vehslin_mdl_callback_mask2_inout.mdl
sl/vehstlin_icon.png
sl/vehstlin_input.m
sl/sim_vehstlin.m
sl/plot_vehslin.m

```
% sim_vehstlin.m
%% Simulate model
mdl = 'vehstlin_mdl_callback_mask2_inout';
% Load model if necessary
if ~bdIsLoaded(mdl)
    open_system(mdl);
end
% Set stop time of simulation according to input
set_param(mdl, 'StopTime', num2str(t_in(end)));
% Save model
save_system(mdl);
% Run simulation
tic;
sim(mdl);
toc;
```

MEGOLDÓK ÉS BEÁLLÍTÁSAIK

- Állandó lépésközű megoldók
  - Az integrálásokat állandó időlépéssel végzik el.
  - Előnyük hogy minden adatsor pontosan reprodukálható, illetve a szimulációs eredmények kiértékelése egyszerűbb lehet.
  - Hátrányuk, hogy túl kicsire választott időlépés esetén a szimuláció lassabb, túl nagyra választott időlépés esetén pedig pontatlanabb lesz.
  - Integrátort nem tartalmazó rendszerben: **`discrete`**
  - Nem merev (non-stiff) megoldók: **`ode1`** , **`ode2`**, **`ode3`**, **`ode4`**, **`ode5`**, **`ode8`**
  - Merev (stiff) megoldó: **`ode14x`**
  - Beállítható a megkívánt fix időlépés (*Fixed-step size*) értéke.
- Változó lépésközű megoldók
  - Az integrálásokat változó, a megoldás lefutásához igazított időlépéssel végzik el.
  - Előnyük, hogy a fejlett és hatékony megoldók általában változó lépésközűek, a szimuláció gyorsabb lehet.
  - Hátrányuk, hogy az egyes szimulációk eredményei egymással közvetlenül nem összehasonlíthatóak, hiszen két jel összehasonlítása csak akkor lehetséges, ha az értékek azonos időpillanatokhoz tartoznak.
  - Integrátort nem tartalmazó rendszerben: **`discrete`**
  - Nem merev (non-stiff) megoldók: **`ode23`**, **`ode45`**, **`ode113`**
  - Merev (stiff) megoldók: **`ode15s`**, **`ode23s`**, **`ode23t`**, **`ode23tb`**
  - Beállítható a javasolt kezdeti időlépés (*Initial step size*), a megengedett minimális (*Min step size*) és maximális (*Max step size*) időlépés, valamint a tolerálható hiba relatív (*Relative tolerance*) és abszolút (*Absolute tolerance*) értéke.

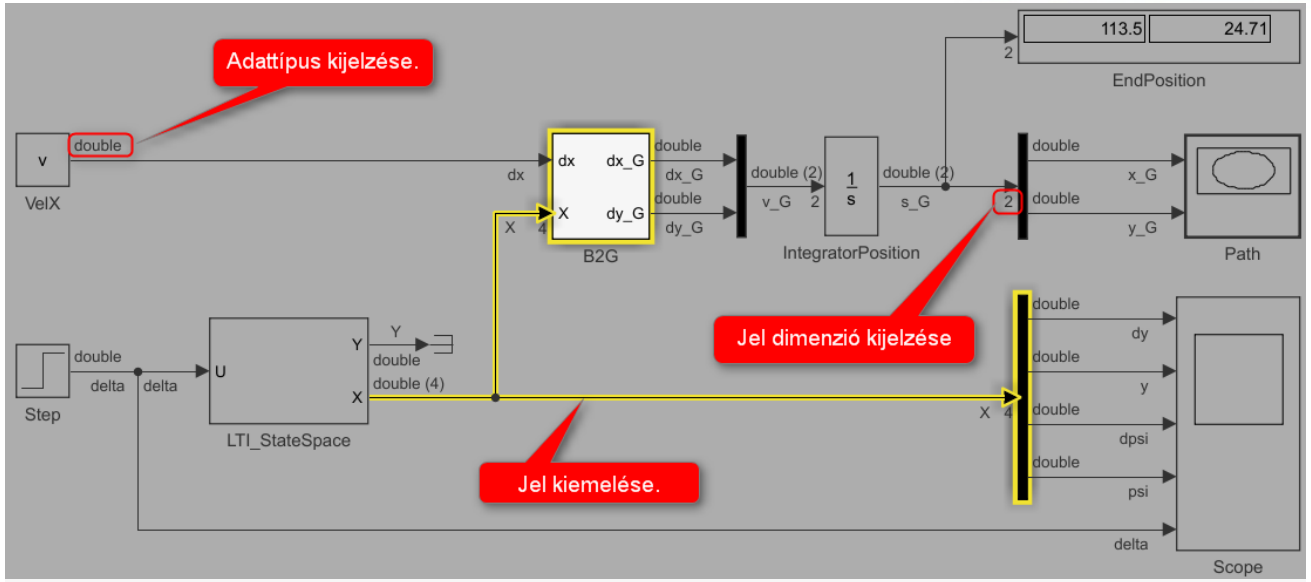
MEGOLDÓK ÉS BEÁLLÍTÁSAIK PÉLDA

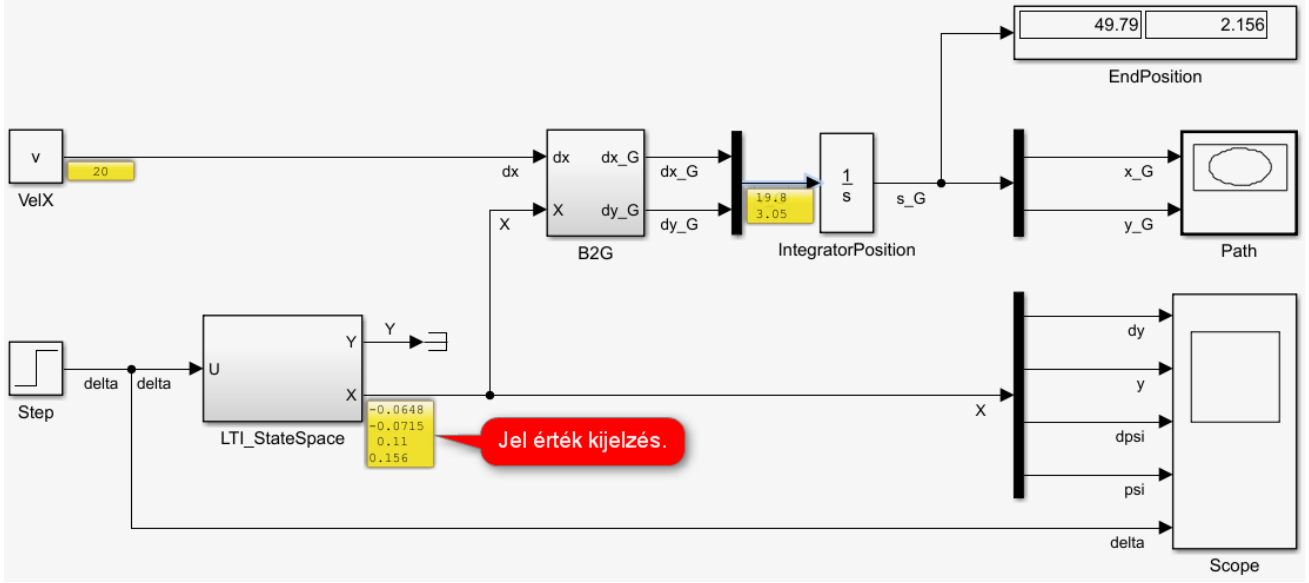


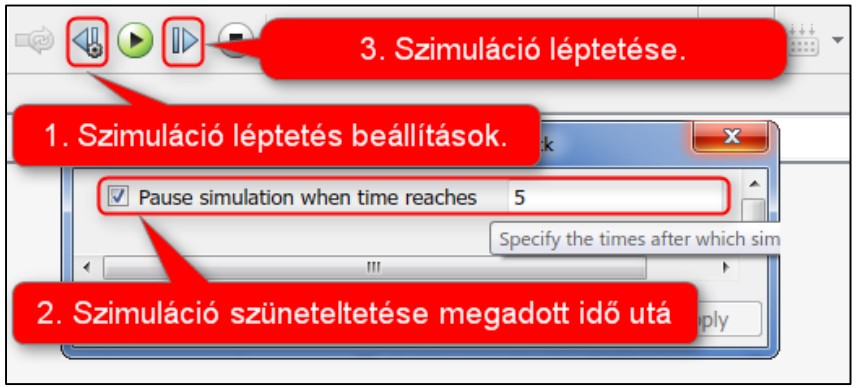
HIBAKERESÉS

- A Simulink modell blokkdiagramján tovább információk jeleníthetőek meg az alapértelmezett nézethez képest:
  - Jel dimenzió: *Display > Signals & Ports > Signal Dimensions*
  - Port adattípus: *Display > Signals & Ports > Port Data Types*
  - Jel értékének kijelzése a jelre klikkelés esetén szimuláció közben: *Display > Data Display in Simulation > Toggle Value Labels When Clicked*
- Egy Simulink jelen jobb egérgombbal kattintva a jel a forrásáig vagy céljáig kiemelhető.
  - *Jobb klikk > Highlight to Destination / Highlight to Source*
  - A kiemelés tetszőleges helyre jobb egérgombbal kattintva és a *Remove Highlighting* opciót választva megszüntethető.
- A Simulink modell szimulációja pillanatnyilag megállítható tetszőleges időpontban, vagy jelértékektől függő töréspontok alapján. A modell futtatása ezután egyszerűen, vagy akár lépésenként is folytatható.
  - A pillanatnyi megállítás időpontja a *Simulation > Stepping options* menüben állítható be.
  - Jelértéktől függő töréspont az adott jelen jobb egérgombbal kattintva és az *Add Conditional Breakpoint* opciót választva állítható be.
  - A modell futtatása a *Simulation > Run / Continue / Step back / Step forward / Stop* menüpontokkal végezhető el.

HIBAKERESÉS PÉLDA







EGYÉB

- Végrehajtási sorrend
  - A Simulink a modellben található függőségek alapján automatikusan meghatározza az egyes blokkok végrehajtási sorrendjét.
  - Ha valami miatt az automatikus végrehajtási sorrendbe bele szeretnénk avatkozni, azt az adott blokkon jobb egérgombbal klikkelve a *Properties* > *General* fülön, a *Priority* mezőben tudunk megadni egy prioritás számot. Ez a szám minél kisebb, a blokk prioritása annál nagyobb, és a blokk annál hamarabb hajtódik végre.
  - Ha a beállított egyéni prioritási sorrend sérti az adatfüggőségeket, akkor hibaüzenetet kapunk.
- Algebrai hurkok
  - Algebrai hurokról beszélünk, ha a modellben van olyan visszacsatolás, melyben csak olyan blokkok vesznek részt, melyek kimenete közvetlenül függ a bemenetüktől.
  - Pl. az alábbi példában a szorzó blokk kimenetének meghatározásához tudnunk kellene mindkét bemenetének az értékét, azonban a **b** bemenet meghatározásához már szükségünk lenne a kimenet ismeretére is.
  - Az algebrai hurkok megoldása problémás, hiszen differenciál-algebrai egyenletrendszerre vezetnek.