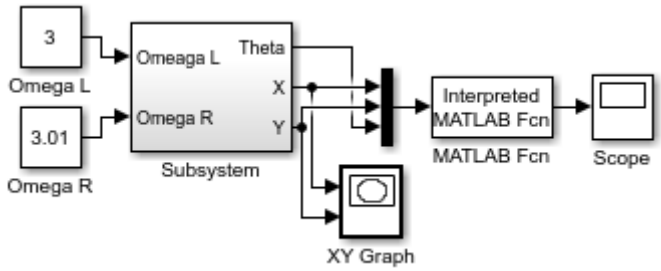
Mechatronika segédlet

10. gyakorlat

2017. április 21.



Vadai Gergely, Faragó Dénes

Tartalom

Ha a jegyzetben bármilyen hibát találsz, kérlek jelezd a farago.denes@stud.u-szeged.hu mailcímen.

Feladatleírás

A gyakorlaton a robot vezérlésének szimulációját valósítjuk meg *Simulink* segítségével. A gyakorlat során használt fájlok az előadó honlapjáról letölthetőek.

simRobot

A `simRobot` függvény először megrajzolja a pályát és a robotot, majd elkészíti a pásztázósugarak mátrixát. A `lineSegmentIntersect` segítségével kiszámítja minden szenzorra, hogyan metszik a pásztázósugarak a pálya vonalait, és ez alapján kiszámítja az ultrahangos szenzorok értékeit.

Paraméterei

`filename`: a pályát tartalmazó *csv* fájl útvonala

(`XP`, `YP`): a robot aktuális tartózkodási helye

`Orient`: a robot aktuális orientációja

`SensorOrients`: a robot szenzorainak orientációja (vektor)

Visszatérési értéke

`sensors`: egy sorvektor, ami a `SensorOrients`nek megfelelő sorrendben tartalmazza az ultrahangos szenzorok értékeit.

Kód

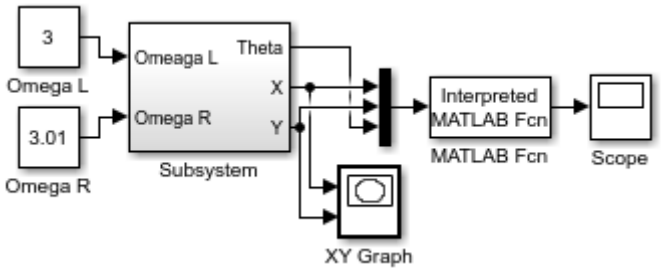
```
function [ sensors ] = simRobot(filename, XP, YP, Orient, SensorOrients,
SensorAngle, SensorRange)
%A függvény kirajzolja a pályát és a robotot, és kiszámítja az ultrahangos
%szenzorok értékeit.
%simRobot('map1.csv', 0, 0, 25, [20, -20, 90, -90, 180], 15, 0.7)

%Kimeneti vektor feltöltése nullákkal
sensors=zeros(size(SensorOrients));

%Rajzoláshoz szükséges vektorok inicializálása:
Xs=[];
Ys=[];

%A felbontás fix:
LineRes=10;
SenRes=11;

%Pálya beolvasása
M=csvread(filename);

%Pálya pontjai:
for i=1:size(M, 1),
    [Xn, Yn]=plotsegment(M(i, 1), M(i, 2), M(i, 3), M(i, 4), LineRes);
    Xs=cat(2, Xs, Xn);
    Ys=cat(2, Ys, Yn);
end

%Szenzorok pontjai:
for i=1:length(SensorOrients),
    [Xn, Yn]=drawarc(XP, YP, SensorRange, SensorAngle,
Orient+SensorOrients(i), SenRes);
    Xs=cat(2, Xs, Xn);
    Ys=cat(2, Ys, Yn);
end
```

```
%Kirajzolás
plot(Xs, Ys, ' .b');
axis equal;

%A pásztázósugarak (X1, Y1) pontjai, vagyis a robot középpontja
%(Kettő darab oszlopvektor)
CenterX=ones(SenRes, 1)*XP;
CenterY=ones(SenRes, 1)*YP;

%Ultrahangos megvalósítás
%Minden szenzorra:
for i=1:length(SensorOrients),

    %Pásztázósugarak végpontjainak meghatározása:
    [Xn, Yn]=drawarc(XP, YP, SensorRange, SensorAngle,
Orient+SensorOrients(i), SenRes);

    %Pásztázósugarak konstruálása a fent definiált Center vektorok
    %segítségével:
    SenRays=[CenterX CenterY Xn' Yn'];

    %Metszéspontok meghatározása
    out = lineSegmentIntersect(SenRays,M);

    %Euklideszi távolságok meghatározása
    Distances=sqrt(((out.intMatrixX-XP).^2)+((out.intMatrixY-YP).^2));

    %Ha volt metszés, akkor az aktuális szenzor értékének kiszámítása
    if (sum(out.intAdjacencyMatrix(:))>0),
        sensors(i)=(min(Distances(out.intAdjacencyMatrix))/SensorRange)*-
1+1;
    end
end
end
```

simRobotMdl

Működése megegyezik a `simRobot`tal azzal a különbséggel, hogy a paraméterek egy részét a *workspace*-ből olvassa be. Ezek a paraméterek függetlenek a *Simulink* modelltől, így az *Interpreted MATLAB Fcn* bemeneteinek száma háromra csökkenthető. Ezeket az értékeket *callback* eljárás segítségével kell értékül adni a *workspacben* lévő megfelelő nevű változóknak. (Lásd: InitFcn)

robotSen.mdl



1. ábra: robotSen

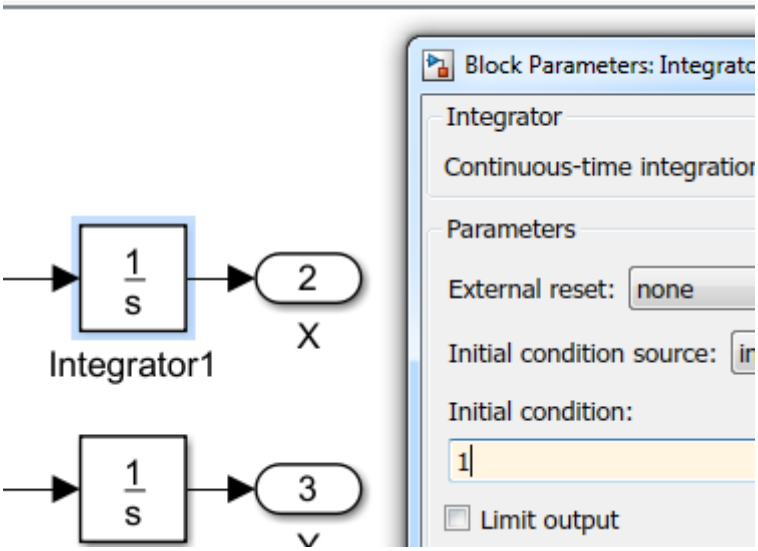
A *robotSen.mdl* a vezérlés, majd később a szabályozás modelljét tartalmazza.

Subsystem

A *Subsytem* a robot matematikai modelljét valósítja meg integrátorok segítségével. A bemenete a differenciál hajtással rendelkező robot bal és jobb oldali kerekének szögsebessége. Kimenete az aktuális pozíció és orientáció.

Integrátorok kezdeti értékei

A *Subsystem*ben lévő integrátorok a rendszer állapotának változásait összegzik, ezért a helyes működéshez meg kell adni a kezdeti pozíciót és orientációt. Ezt az integrátorok kezdőértékeinek megadásával tehetjük meg: *dupla kattintás az integrátorra -> properties -> initial condition* (2. ábra).



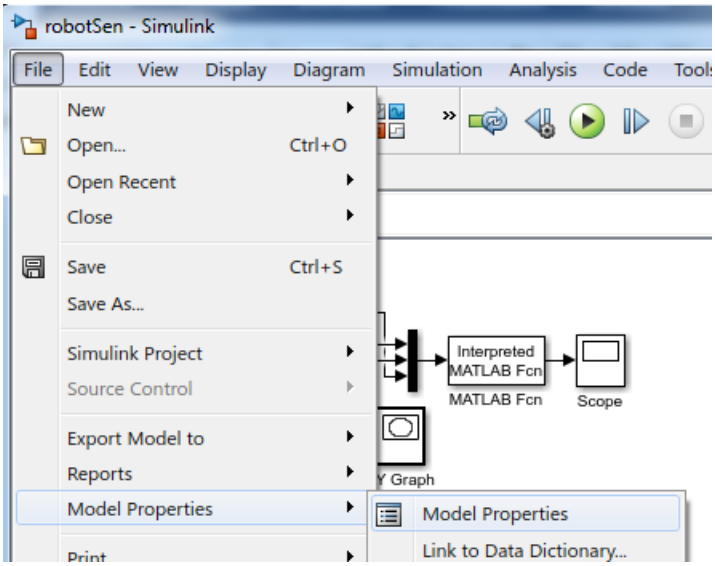
2. ábra: Az integrátorok kezdőértékeinek beállítása. Ebben az esetben a robot kezdeti X koordinátája 1.

Interpreted MATLAB Fcn

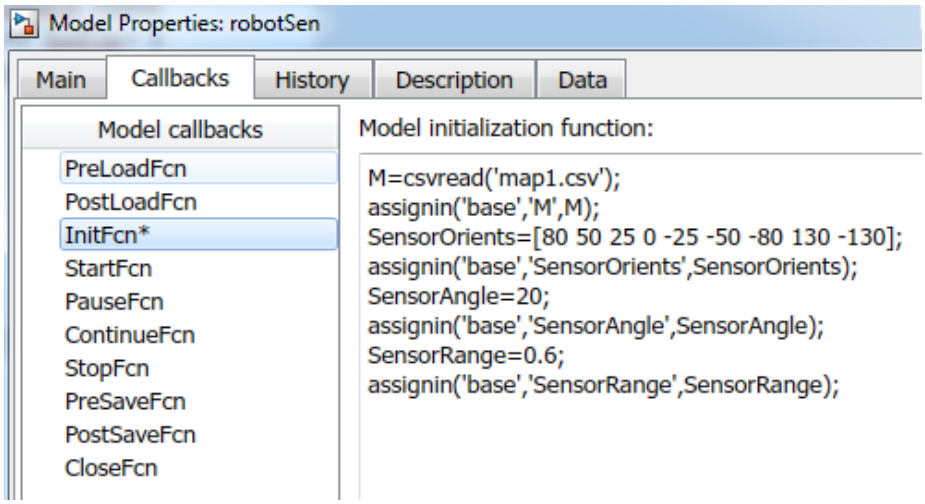
Ez a blokk meghívja a *simRobotMdl* függvényt. A *Subsystem* kimeneti értékei egy multiplexeren keresztül a *simRobotMdl* függvénybe jutnak, amely minden iterációban kirajzolja a robotot és a pályát, valamint kiszámítja a szenzorok értékeit.

InitFcn

A `simRobotMdl` a paramétereinek egy részét a közös *workspace*-ről olvassa be, ezért a szimulációk előtt ezeket létre kell hozni a megfelelő változónevekkel. Ezt megtehetjük a modellhez tartozó *InitFcn callback* eljárás segítségével is, ami akkor hívódik meg, amikor a szimulációt elindítjuk. Ennek módosításához modellen belül a *File -> Model Properties -> Model Properties*-re kell kattintani, azon belül pedig a *Callbacks* fület kell kiválasztani és az *InitFcn* menüpontot.



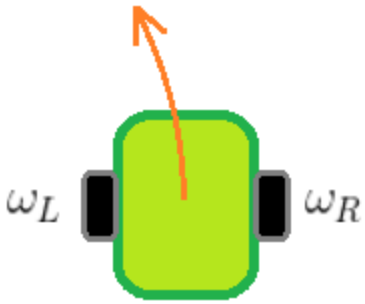
3. ábra: Model Properties ablak elérése



4. ábra: Szimuláció előtt végrehajtódó eljárás megadása

Differenciál hajtású robot vezérlése

A differenciál hajtású robotok irányítása a kerekek szögsebességének változtatásával valósul meg. A mi modellünkben ez az irányítás jelenleg konstansokkal van megvalósítva, a következő órán ezt helyettesítjük egy P-típusú szabályzással. Jelenleg a bal oldali kerék sebessége kisebb, mint a jobb oldali keréké, ezért a robot az óramutató járásával ellentétes irányban fog körözni (5. ábra).



5. ábra: Differenciál hajtással rendelkező robot. Az ábrán a bal oldali kerék lassabban forog, mint a másik.