

Kódgenerálás és GUI fejlesztés Matlab-Simulink környezetben

A mérés célja: Grafikus felhatalnáló felülettel rendelkező alkalmazás létrehozása a Matlab alatt, az App Designer eszköz és elemkönyvtárainak megismerése, kapcsolat a Simulink modellekkel, szimuláció vezérlése.

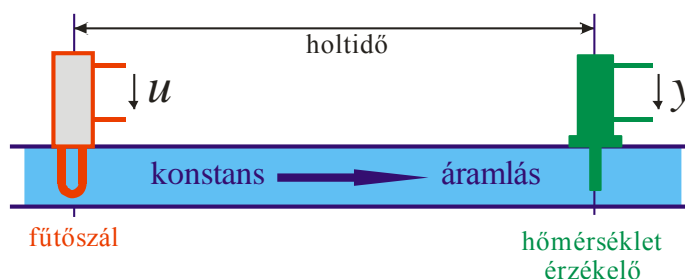
Felhasznált eszközök: (IK Cloud architektúra – <http://cloud.ik.bme.hu>), Matlab R2017a vagy 2017b, Simulink, App Designer, Matlab Compiler.

Rendelkezésre álló, letölthető állományok:

- Mérési útmutató (jelen leírás)
- [Holtidos szakasz.jpg](#) (szakasz működését szemléltető ábra)

1. feladat – Egyszerű alkalmazás grafikus felülettel

A feladat grafikus felhatalnáló felület (GUI) létrehozása, amely támogatja, hogy egy holtidős, egytárolós szakaszhoz PI szabályzó méretezést végezzünk¹. A PI szabályzót egy fűtött folyadékot keringető berendezés (például dialízisgép) hőmérséklet szabályozójához kell megtervezni, amelynek sémája az ábrán látható.



1. ábra. A holtidős szakasz működésének vázlata.

A szakasz viselkedése egy adott hőmérséklet, mint munkaponti érték környékén jól jellemezhető egy átviteli függvénnyel: $W(s) = \frac{A}{1+sT} \exp(-sT_h)$. Ehhez az átviteli függvényhez tervezünk egy $W_{PI}(s) = \frac{A_p}{T_i} \frac{1+sT_i}{s}$ átvitelű PI szabályzót.

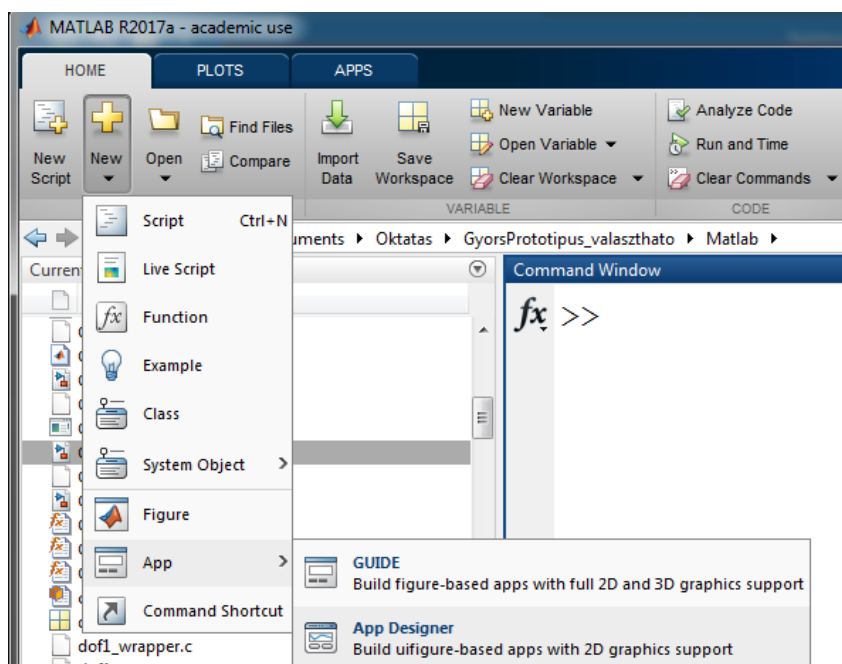
Megoldás: A fejlesztést a GUIDE vagy az App Designer eszköz segítségével végezhetjük. A GUIDE összetettebb eszköz, ugyanakkor egyszerűbb applikációk (Appok) fejlesztéséhez az egyszerűsített és letisztultabb App Designer javasolható. A feladatban szereplő eszköz esetében az App Designer alkalmazása elegendő lesz, az alkalmazásmodell a GUIDE esetében is hasonló.

A fejlesztő eszköz indításakor egy újabb felület jelenik meg. Az alkalmazás felépítése egyszerű. Külön szerkesztői nézetben helyezhetjük el a felhatalnáló felület elemeit (Design View). Az programozási modell eseményvezérelt, az egyes kezelőelemek eseményeihez rendelt kódot is egy külön nézetben szerkeszthetjük (Code View). A kód egy részét a fejlesztőkörnyezet generálja, ennek szerkesztése nem tanácsos.

A fejlesztőkörnyezet további elemeinek használata is értelemeszerű. Az alkalmazható kezelőszervek könyvtára a bal oldalon található, ezeket *drag & drop* művelettel példányosíthatjuk az alkalmazásunkban. Az egyes elemekhez

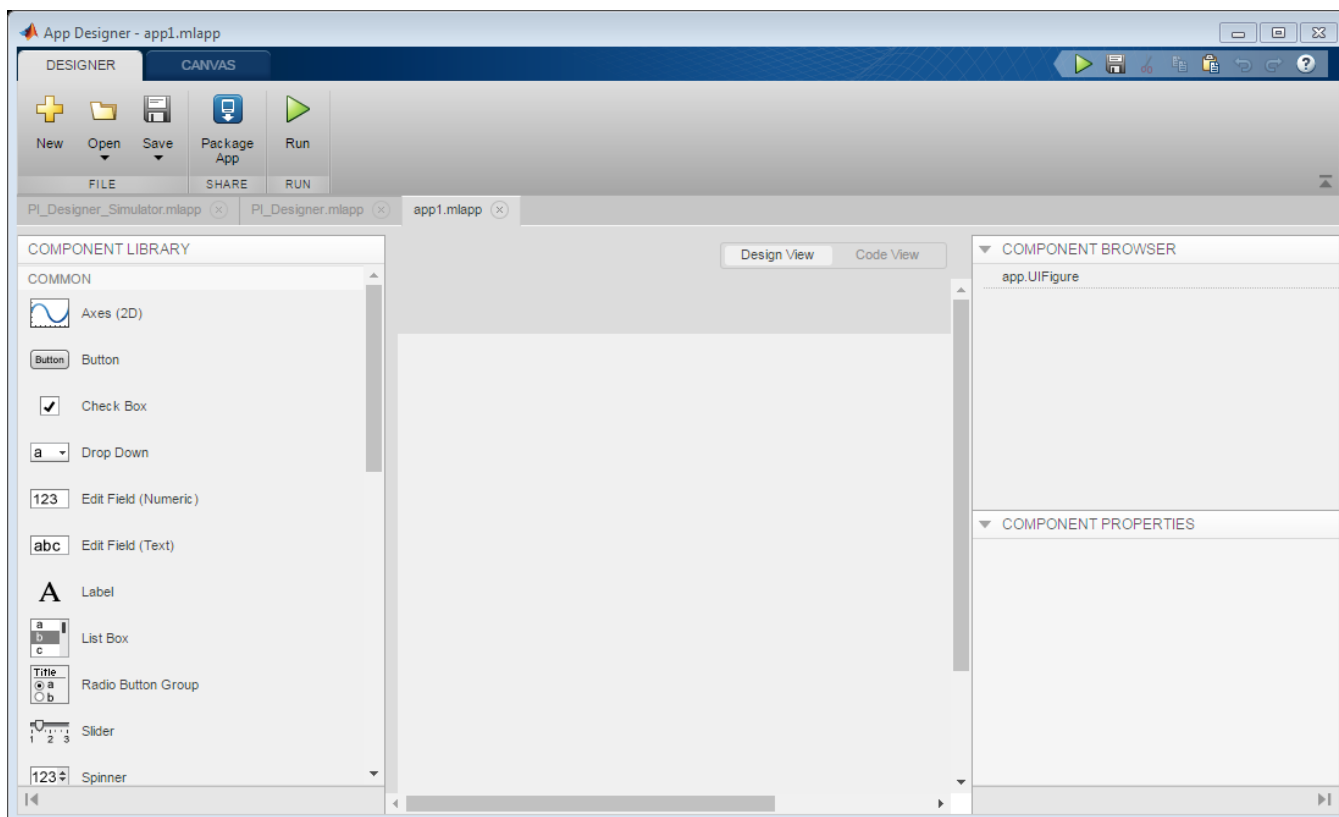
¹ PI szabályzó méretezése a Rendszerelmélet (VIHVAB00) tárgyból szerepelt

eseménykezelő függvényeket a jobb oldali menüben rendelhetünk és itt állíthatók a felület elemeinek tulajdonságai is.



2. ábra. Alkalmazások fejlesztése (GUIDE és App Designer)

Az alkalmazást a Run paranccsal indíthatjuk, a futtatás értelemszerűen a Matlab keretrendszerben zajlik.



3. ábra. Az App Designer felülete.

Kezdjük a kép elhelyezésével. Képet a `Axes (2D)`, `UIAxes` osztályú objektum esetében mutathatunk a felhasználónak. A `jpg` képet az erre szolgáló `imshow` metódus meghívásával lehet beilleszteni. A metódus hívását ugyanakkor be kell illeszteni egy eseménykezelő függvénybe, viszont az `UIAxes` objektumokhoz nem tartozik eseménykezelő. Ezért az alkalmazás ablakának (`UIFigure`) egyik eseménykezelőjét használjuk

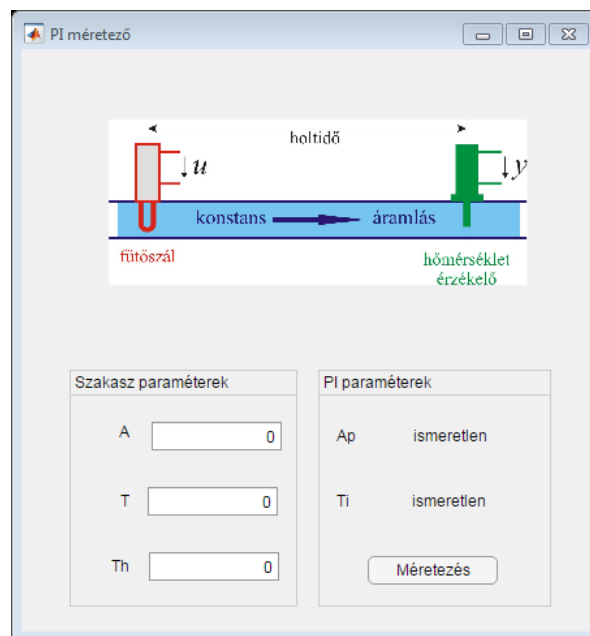
(`startupFcn`), amely annak létrehozásához tartozó esemény nyomán kerül végrehajtásra. Ennek kódját mutatja a 4. ábra.

```
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
    imshow('Holtidos_szakas.jpg', 'parent', app.UIAxes);
end
```

4. ábra. Kép mutató UIAxes osztályú objektummal.

A PI szabályozó tervezéséhez a fejlesztői felületen például a 5. ábra szerinti módon helyezhetjük el az elemeket. A szakasz paramétereinek megadásához numerikus szerkesztő mezőket (Edit Field (Numeric)), a számított paraméterek megjelenítéséhez pedig Label elemeket használhatunk.



5. ábra. A PI méretező felülete.

A tervező eljárást egy különálló függvényben is lehet implementálni, hogy a fázistartalék például 60 fok. A méretezés a korábban tanultak alapján mindössze három sor, amit a felület Méretezés nyomógombjának eseményéhez rendelünk.

```
% Button pushed function: DesignButton
function DesignButtonPushed(app, event)
    A = app.PlantGain.Value;
    T = app.PlantTimeCte.Value;
    Th = app.PlantTimeLag.Value;
    % controller design calculations for 60 degrees phase margin
    Ti = T;
    wc = pi/6/Th;
    Ap = Ti*wc/A;
    app.TiValue.Text = num2str(Ti);
    app.ApValue.Text = num2str(Ap);
end
```

6. ábra. A Méretezés gomb megnyomásához tartozó eseménykezelő függvény kódja.

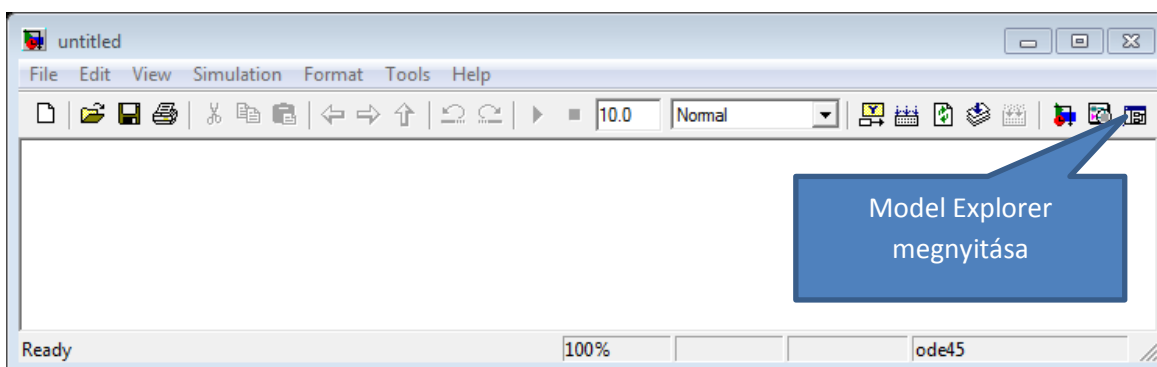
2. feladat – A PI méretező módosítása

Módosítsa az előző feladat nyomán keletkezett alkalmazást, hogy a fázisstartalék értékét is a felhasználó adhassa meg!

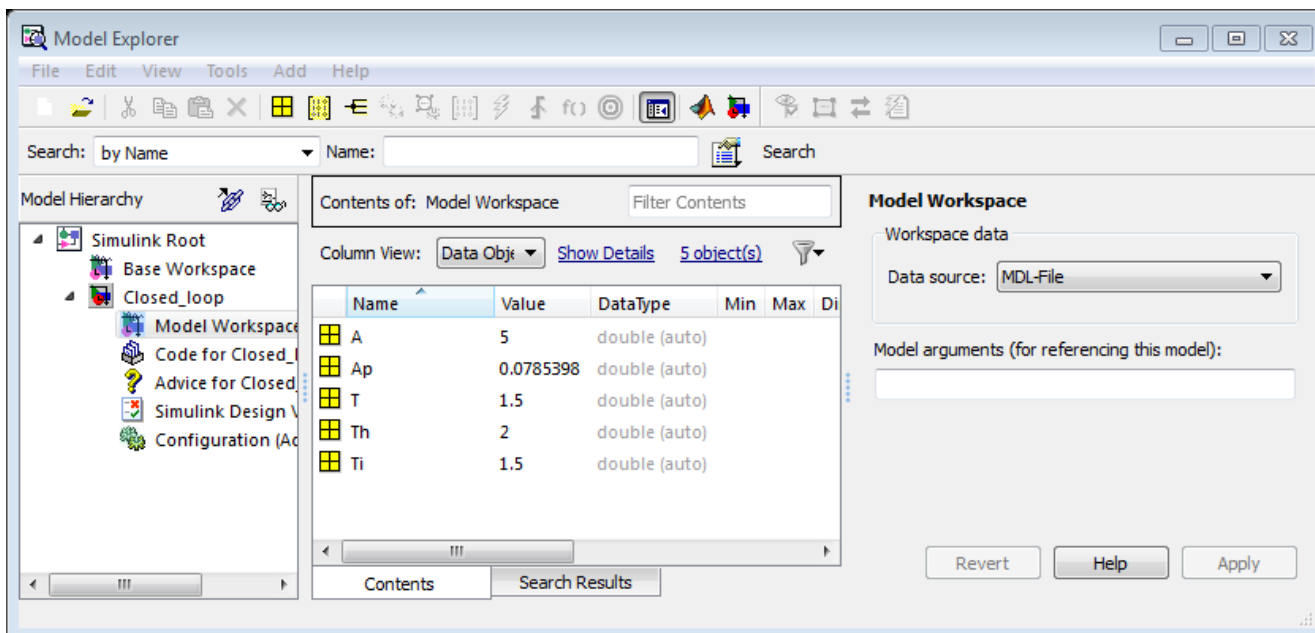
3. feladat – A PI méretező tesztelése szimulációval

Hozzon létre egy Simulink modellt a zárt szabályozási kör szimulációjához, amelynek munkaterében szerepelnek változóként a szakasz és a szabályzó paraméterei! Egészítse ki a felhasználói felületet, hogy a szabályzó méretezése nyomán a zárt kör szimulációjának eredménye a felületen megjelenjen (egységugrás alapjel esetén).

Megoldás: A Matlabon belül nem csak az 'alap' munkatér (Base Workspace) tartalmazhat változókat, minden Simulink modellnek is van saját munkatere. A szabályozási körünk paramétereit a modell munkaterében hozzuk létre a modellbongésző (Model Explorer) segítségével. Ehhez először egy modellt kell megnyitni.

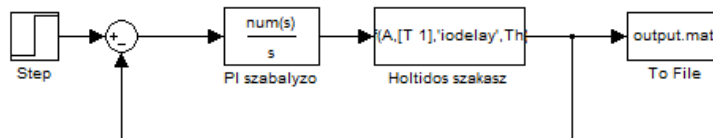


A modellbongészőben a model munkaterében egyszerűen létrehozhatjuk és inicializálhatjuk a változókat az összes paraméter számára (szakasz és szabályzó). A változókra a modell blokkjainak párbeszéd ablakában hivatkozhatunk.



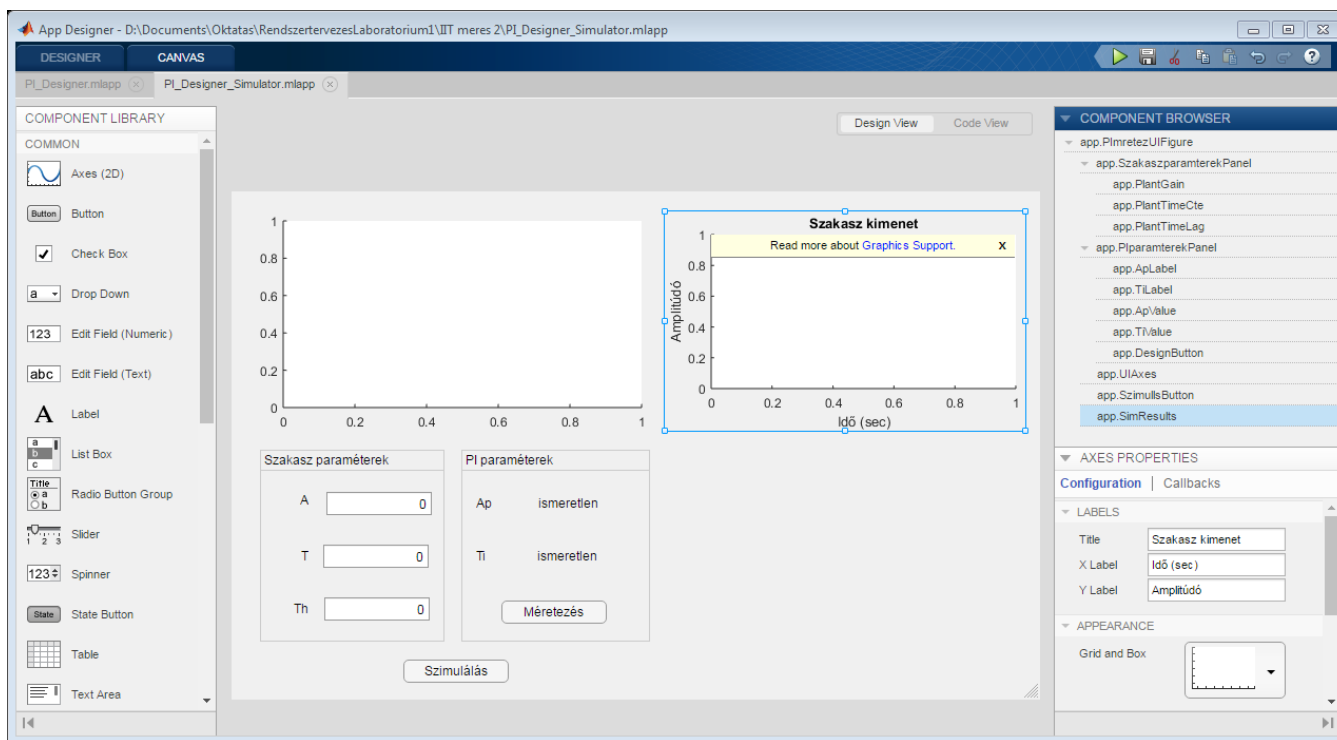
7. ábra. A modell böngésző ablaka

A Simulink modellben a holtidőt is kezelő LTI blokkot használunk és a szimulált kimenetet egy `mat` fájlba mentjük.



8. ábra. A holtidőt tartalmazó szabályozási kör Simulink-ben

A felhasználói felületet ki kell egészíteni egy `UIAxes` típusú objektummal, amelyben a szimulált tranzienszt fogjuk megjeleníteni. Természetesen ezt megintcsak a `App Designer` segítségével tesszük meg.



9. ábra. Újabb `UIAxes` objektum hozzáadása a felülethez a szimuláció eredményének megjelenítéséhez

Utolsó lépésként a szimulációhoz tartozó gomb eseménykezelőjének megadására van szükség, amely az alábbi feladatokat végzi el:

- módosítja a modell munkatérében található változókat
- lefuttatja a szimulációt
- betölti az eredményeket tartalmazó állományt
- törli az aktuális tranzienszt és a helyére kirajzolja az újat

Ehhez az alábbi utasítások állnak rendelkezésünkre:

- Simulink modellünk megnyitására a `load_system` utasítás szolgál
- A munkatérhez tartozó `handle`-t a `get_param` utasítás adja vissza
- Valamely munkatér változójához az `assignin` utasítással férhetünk hozzá
- Modellünk szimulációját az `sim` utasítással kezdeményezhetjük

A keletkező kódot a 10. ábra mutatja.

```

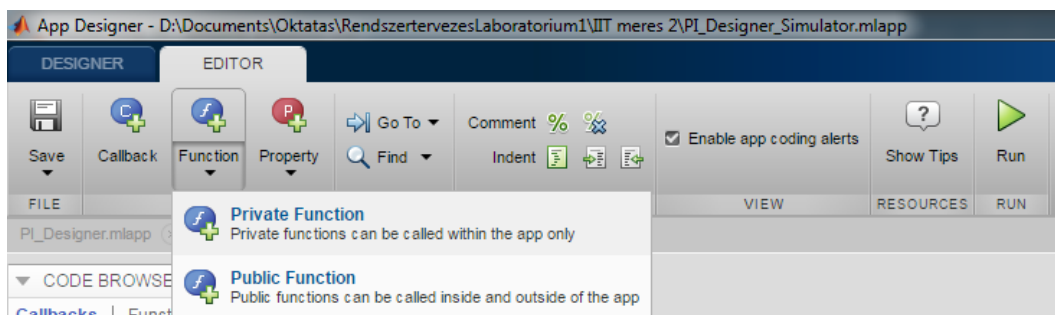
% Button pushed function: SzimullButton
function SzimullButtonPushed(app, event)
    load_system('Closed_loop_PI');
    hws = get_param(bdroot,'modelworkspace');
    hws.assignin('A',app.PlantGain.Value);
    hws.assignin('T',app.PlantTimeCte.Value);
    hws.assignin('Th',app.PlantTimeLag.Value);
    hws.assignin('Ap',str2num(app.ApValue.Text));
    hws.assignin('Ti',str2num(app.TiValue.Text));
    save_system;
    sim('Closed_loop_PI');
    load output;
    plot(app.SimResults,y(1,:),y(2,:));
end

```

10. ábra. Szimuláció futtatása eseménykezelő függvényből.

4. feladat – Az alkalmazás kiegészítése

1. Egészítse ki az előző feladatban kapott alkalmazást, hogy a szimuláció időtartalma mindig automatikusan a beállított holtidő ötszöröse legyen! A szimulációs idő a Simulink modell egyik paramétere és közvetlenül beállítható a `sim` utasítás egy argumentumaként (v.ö. `doc sim`).
2. Egészítse ki az előző feladatban kapott alkalmazást, hogy a felületen a beavatkozó jel is megjelenjen!
3. Egészítse ki az alkalmazást, hogy a szabályozó paramétereit egy belső (`private` láthatósági osztályú) függvény számolja. Ilyen függvény létrehozásához nyújt segítséget a 11. ábra.



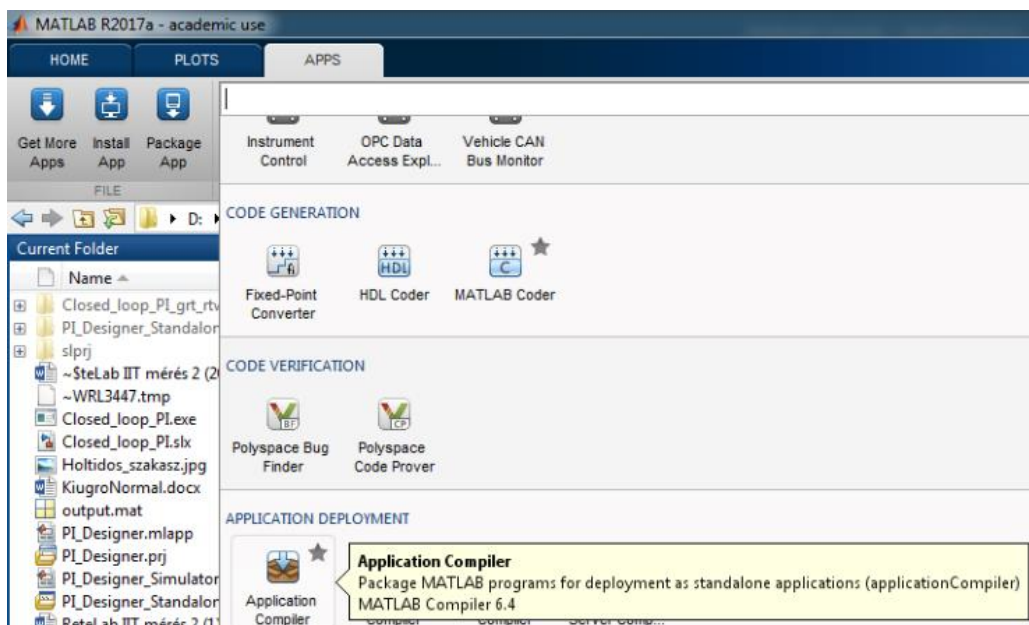
11. ábra. Függvények (`private` vagy `public` láthatósági osztályú) létrehozása

5. feladat – Önálló alkalmazás létrehozása az elkészített App-ból

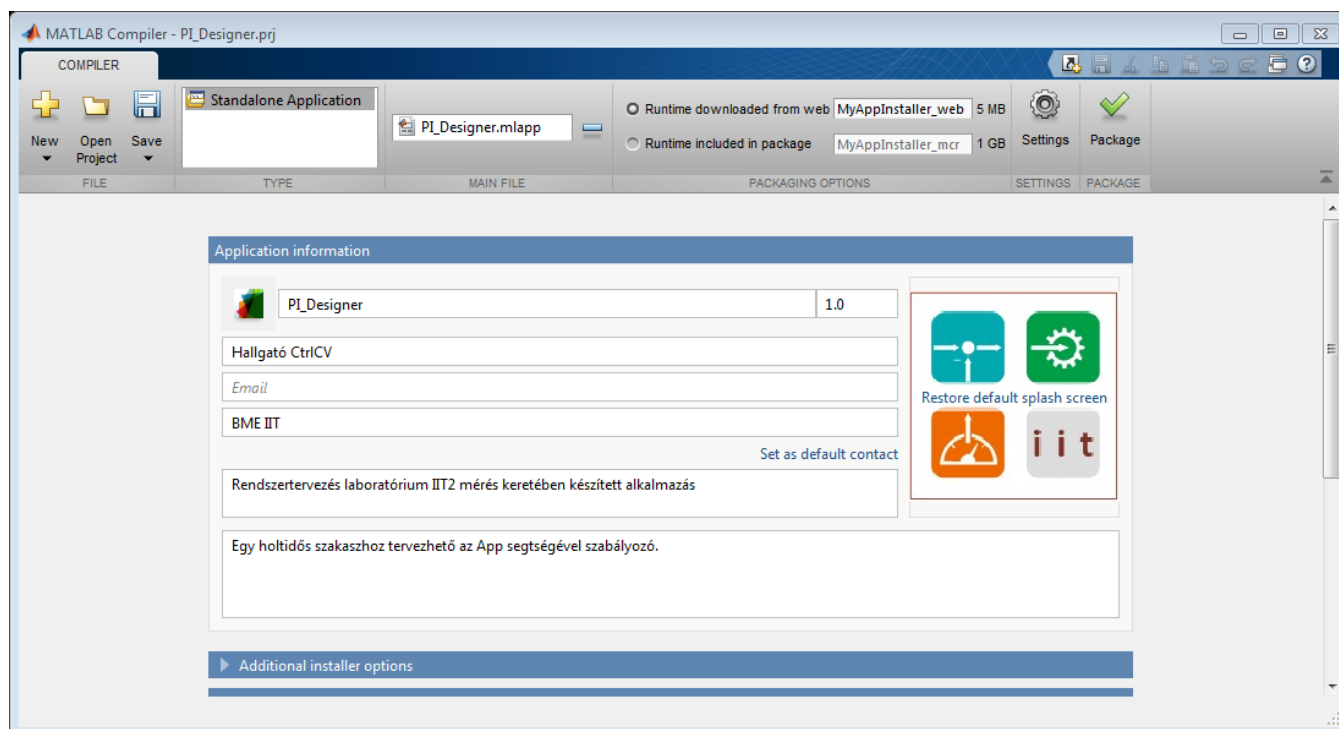
Az eddig elkészült Appok futtatásához a Matlab indítása szükséges volt. Ugyanakkor a Matlab fordítóval (Matlab Compiler) eszközzel segítségével lehetőség van olyan önálló (standalone) alkalmazás előállítására is, amelynek futtatásához nincsen szükség a Matlab indítására. Fontos megjegyezni ugyanakkor, hogy a lefordított alkalmazáshoz vagy hozzácsomagoljuk a Matlab egy részét (ez akár 1Gb is lehet) vagy lehetővé tesszük, hogy ehhez az alkalmazásunk a hálózaton keresztül férjen hozzá. A fordító is egy App persze, ezt mutatja a 12. ábra.

A felület használata értelemszerű, esetünkben a szükséges állományok kiválasztása is egyszerű. A kiválasztandó `MAIN FILE` esetünkben a `PI_Designer.mlapp`, további függvényekre és szkriptekre nincsen szükség. Az alkalmazás létrehozásához a `Package` parancsot kell használni. Sajnos a kód analízise nem elég fejlett ahhoz, hogy az `imshow` utasításnál megadott `Holtidos_szakas.jpg` fájlt is a csomagolni kívánt állományok közé illessze a keretrendszer, így azt kézzel kell megtenni a `Files installed for your end user` mezőnél.

A hosszas fordítás után a végrehajtható állományok a `PI_Designer` könyvtárban keletkeznek. A Matlab három különböző változatot is előállít, próbálkozzunk a `for_redistribution_files_only` könyvtárban található `PI_Designer.exe` futtatásával.



12. ábra. A Matlab fordító felületének indítása



13. ábra. A Matlab fordító felülete