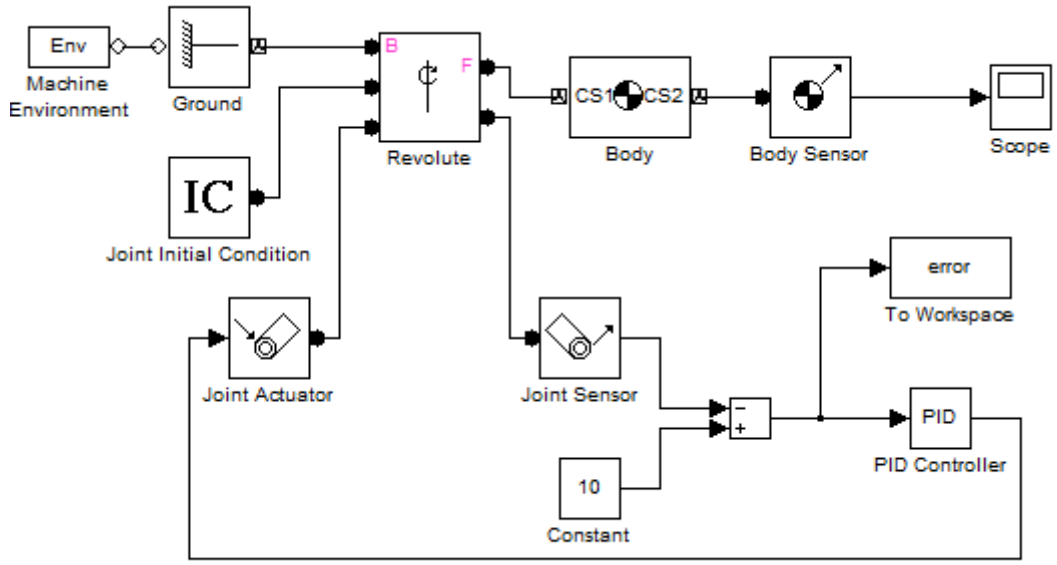
Intelligens rendszerek gyakorlat

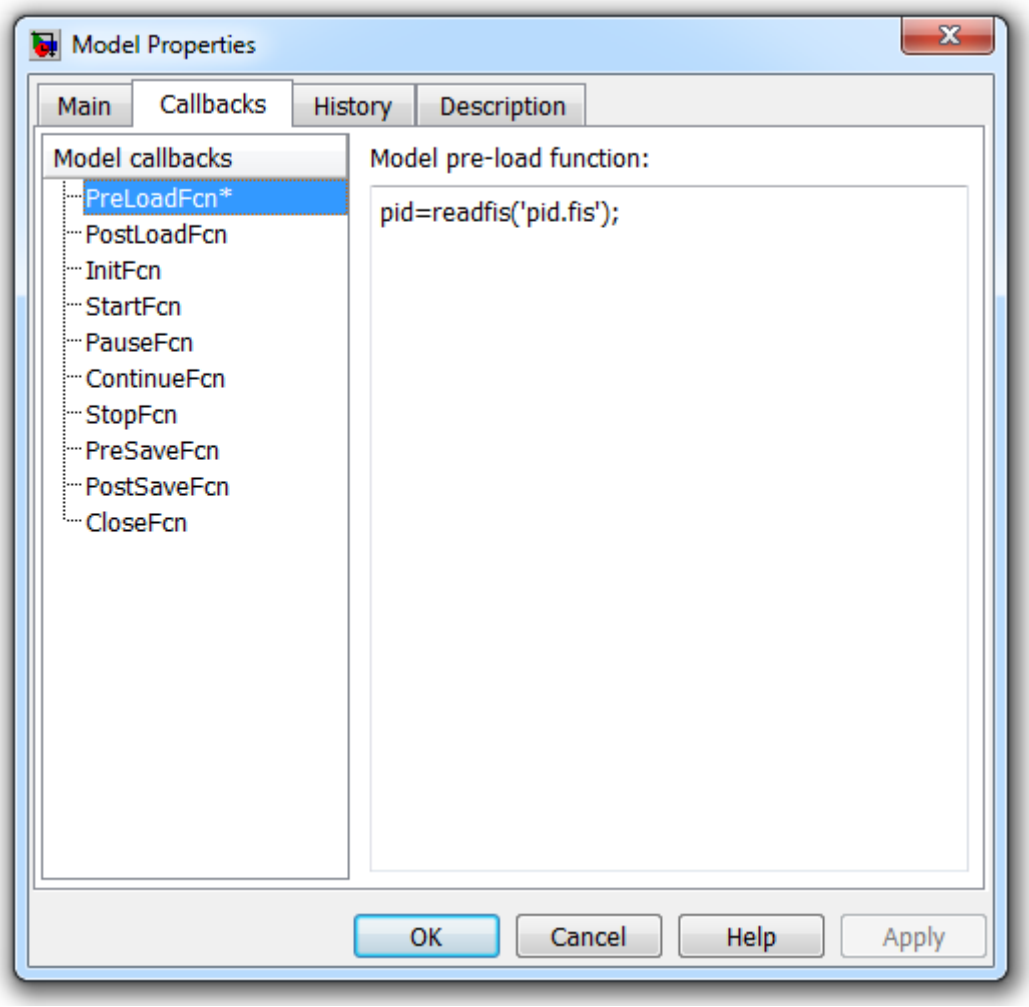
A mai gyakorlat célja hogy megnézzük, hogyan lehet a genetikus algoritmussal behangolni a PID paramétereket.

Töltsük le a www.inf.u-szeged.hu/~szepet oldalról a Pendulum Optimisation linkről a pendopt.zip-et. Ezt csomagoljuk ki a MATLAB által elérhető mappába.

Ez a zip két fájlt tartalmaz. Az egyik a pend.m míg a másik a pendopt.mdl. Nyissuk meg a pendopt.mdl-t.



A fájl megnyitásakor egy hibaüzentet kaphatunk. Kattintsunk jobb egér gombbal a modell fehér részére és válasszuk a Modell Properties-t. Ekkor az alábbi ablak nyílik meg. Itt a második fület kiválasztva látható, hogy van egy PreLoadFcn. Mivel most nincs ilyen pid.fis fájlunk ez okozta a hibaüzenetet.



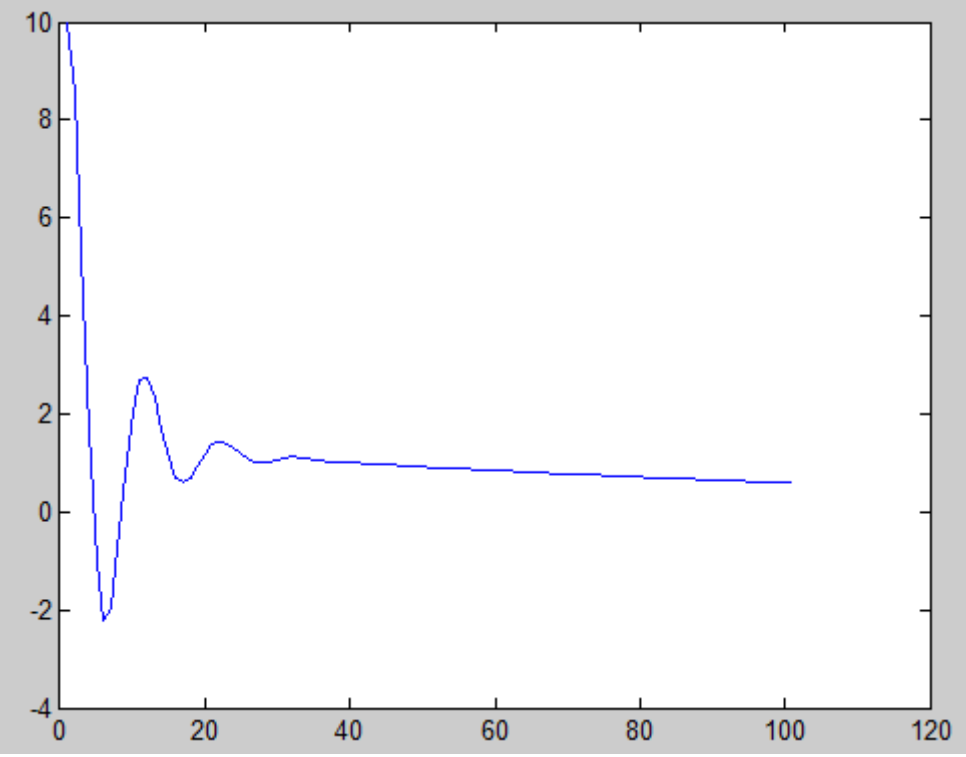
Egyszerűen töröljük ki ezt a sort, és mentsük el a modellt. Most nyissuk meg a pend.m fájlt, melynek a tartalma a következő:

```
function [] = pend(pars)
%input paraméterek workspace-be töltése
assignin('base', 'p_val', pars(1));
assignin('base', 'i_val', pars(2));
assignin('base', 'd_val', pars(3));

%modell definíció
mdl='pendopt.mdl';
%modell betöltése
load_system(mdl);
%modell szimulálása 0-tól 10 másodpercig a base workspace-ben
sim(mdl, 0:10, simset('DstWorkspace', 'base'));

%a szimuláció toworkspace doboza által visszaadott jelsorának
%kinyerése a base workspace-ből
y=evalin('base', 'error.signals.values');

%kinyert jelsor plottolása
plot(y);
end
```

Ez egy példa arra hogy hogyan lehet MATLAB-ból futtatni egy Simulink modellt. Futtassuk is le ezt az m function-t.

```
>> pend([1 0.1 0.1])
```

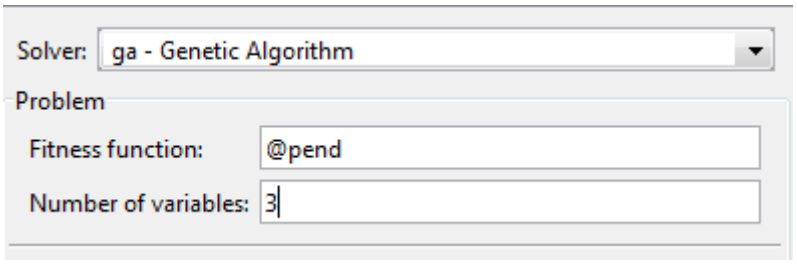
Ekkor az alábbi eredmény kapjuk:

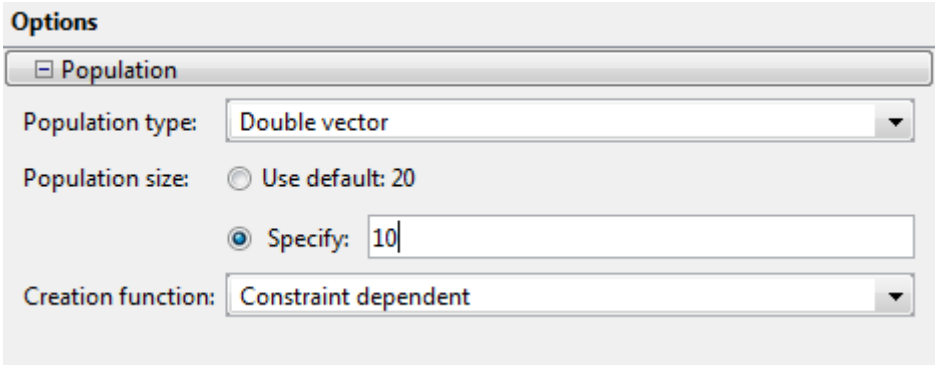


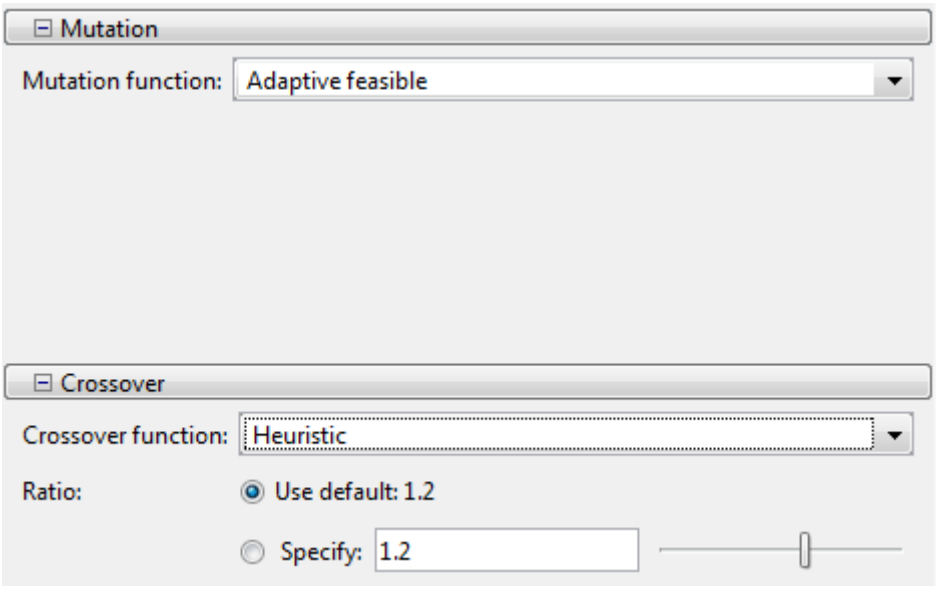
A modellből a hibajel van kivezetve és ezt plot-oltuk ki. Látható hogy a hiba 0-ra csökkent. Próbáljuk meg a genetikus algoritmussal beállítani a PID paramétereket. Ehhez először is ki kell egészíteni a pend.m fájlt az alábbi módon:

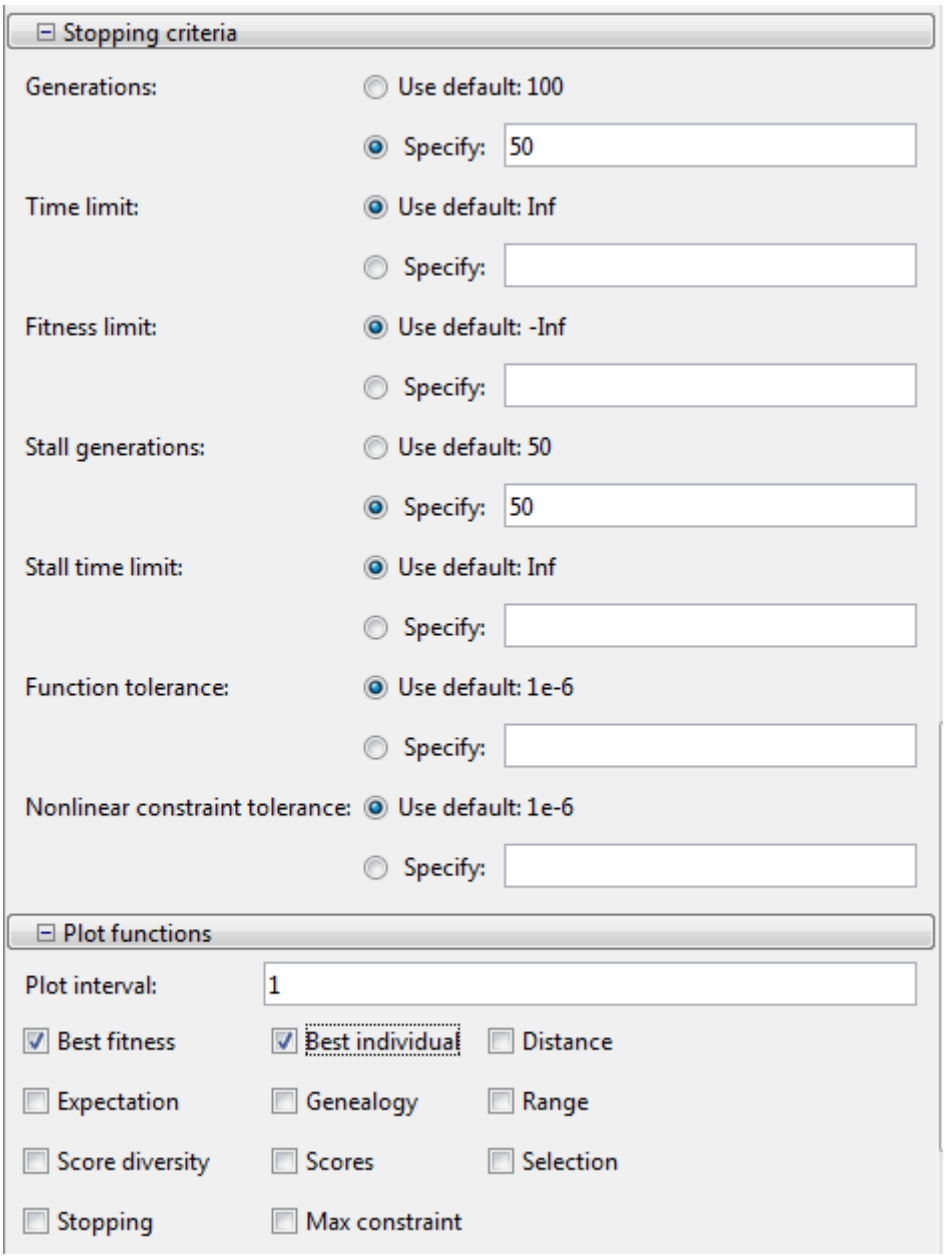
```
function [ret] = pend(pars)
%input paraméterek workspace-be töltése
assignin('base', 'p_val', pars(1));
assignin('base', 'i_val', pars(2));
assignin('base', 'd_val', pars(3));

%modell definíció
mdl='pendopt.mdl';
%modell betöltése
load_system(mdl);
%modell szimulálása 0-tól 10 másodpercig a base workspace-ben
sim(mdl, 0:10, simset('DstWorkspace', 'base'));

%a szimuláció toworkspace doboza által visszaadott jelsorának
%kinyerése a base workspace-ből
y=evalin('base', 'error.signals.values');

%kinyert jelsor plottolása
%plot(y);
ret=sum(y);
end
```

Készítettünk a függvénynek egy visszatérési értéket. Indítsuk el az optimtoolt-t és próbáljunk meg a genetikus algoritmussal optimális PID paramétereket találni.

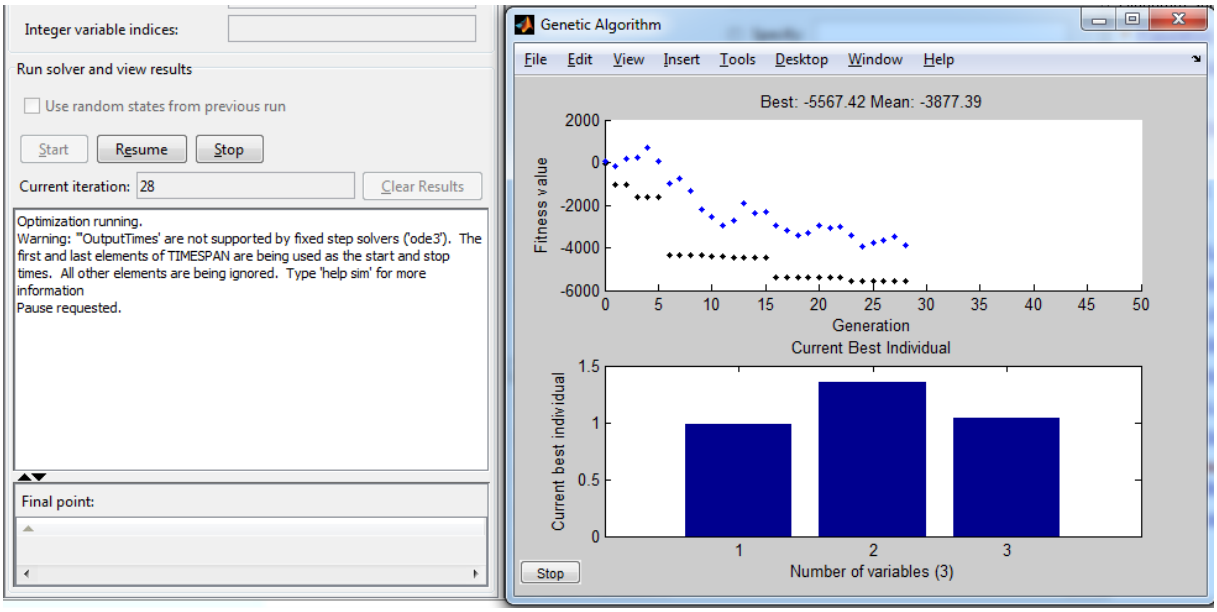





Mutation
Mutation function:
Adaptive feasible
Crossover
Crossover function:
Heuristic
Ratio:
Use default: 1.2
Specify:
1.2


Stopping criteria
Generations:
Use default: 100
Specify:
50
Time limit:
Use default: Inf
Specify:
Fitness limit:
Use default: -Inf
Specify:
Stall generations:
Use default: 50
Specify:
50
Stall time limit:
Use default: Inf
Specify:
Function tolerance:
Use default: 1e-6
Specify:
Nonlinear constraint tolerance:
Use default: 1e-6
Specify:
Plot functions
Plot interval:
1
Best fitness
Best individual
Distance
Expectation
Genealogy
Range
Score diversity
Scores
Selection
Stopping
Max constraint

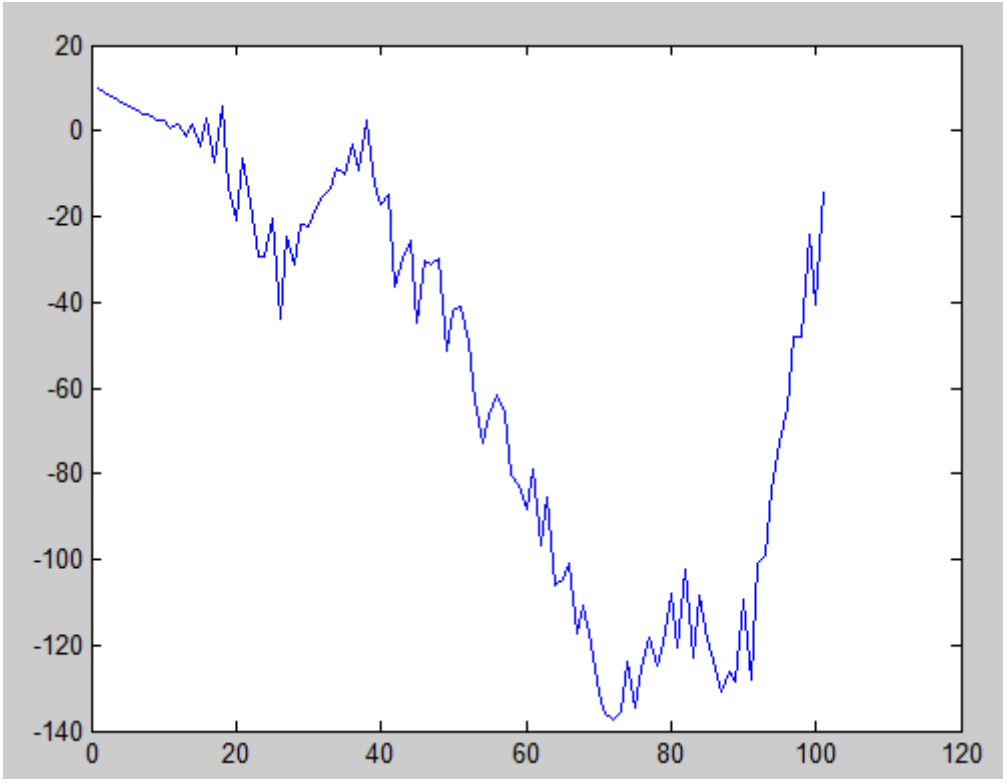
Látszik, hogy a fittness függvény által adott kimenet folyamatosan nő negatív irányba.



Állítsuk meg az algoritmust és nézzük meg mit kapunk a megadott paraméterekkel.

```
>> pend([0.9866964086218089 1.3513438462559615  1.04333367173229 52])
```

```
>> plot(error.signals.values)
```



Látszik, hogy a meghatározott paraméterekkel teljesen rosszul működik a szabályzás. Mi lehet ennek az oka?

Ha megnézzük az eredeti 1, 0.1, 0.1 paraméterekkel a futtatás során kapott hibát:

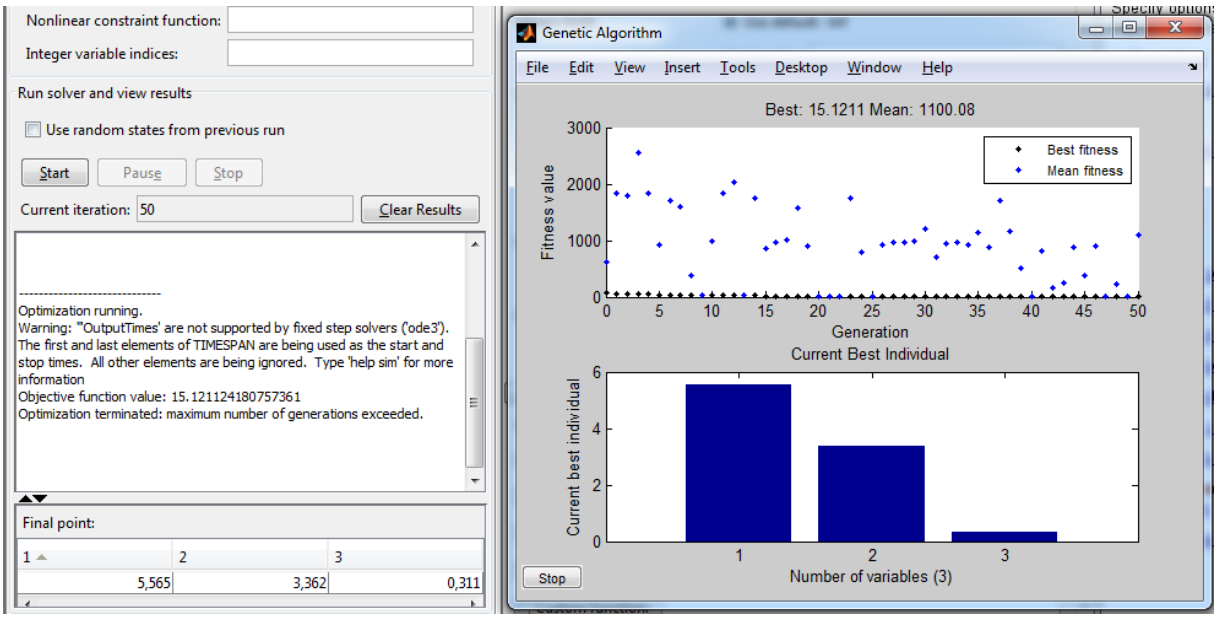
```
>> y=error.signal

y =

   10.0000
    8.5628
    5.2700
    1.6761
   -1.0124
   -2.2109
   -1.9649
   -0.7585
    0.7588
    2.0218
    2.7027
    2.7475
    2.3166
```

Láthatjuk hogy vannak benne negatív értékek, és mi azt implementáltuk a jósági függvényre, hogy adja ezeket az értékeket össze, azaz integráljuk a hibajelet. Ha egy előjeles adatsort integrálunk és mi genetikus algoritmust használunk, akkor az algoritmus hamar rá fog jönni arra, hogy az lesz a cél, hogy az idő legnagyobb részében a hibajel negatív legyen és minél nagyobb.

Nem korlátos a fittness function alulról. Tegyük alulról korlátossá a függvényt, azaz vegyük a hibajel abszolút értékét és így is próbáljunk optimalizálni.

```
function [ret] = pend(pars)
%input paraméterek workspace-be töltése
assignin('base', 'p_val', pars(1));
assignin('base', 'i_val', pars(2));
assignin('base', 'd_val', pars(3));

%modell definíció
mdl='pendopt.mdl';
%modell betöltése
load_system(mdl);
%modell szimulálása 0-tól 10 másodpercig a base workspace-ben
sim(mdl, 0:10, simset('DstWorkspace', 'base'));

%a szimuláció toworkspace doboza által visszaadott jelsorának
%kinyerése a base workspace-ből
y=evalin('base', 'error.signals.values');

%kinyert jelsor plottolása
%plot(y);
ret=sum(abs(y));
end
```

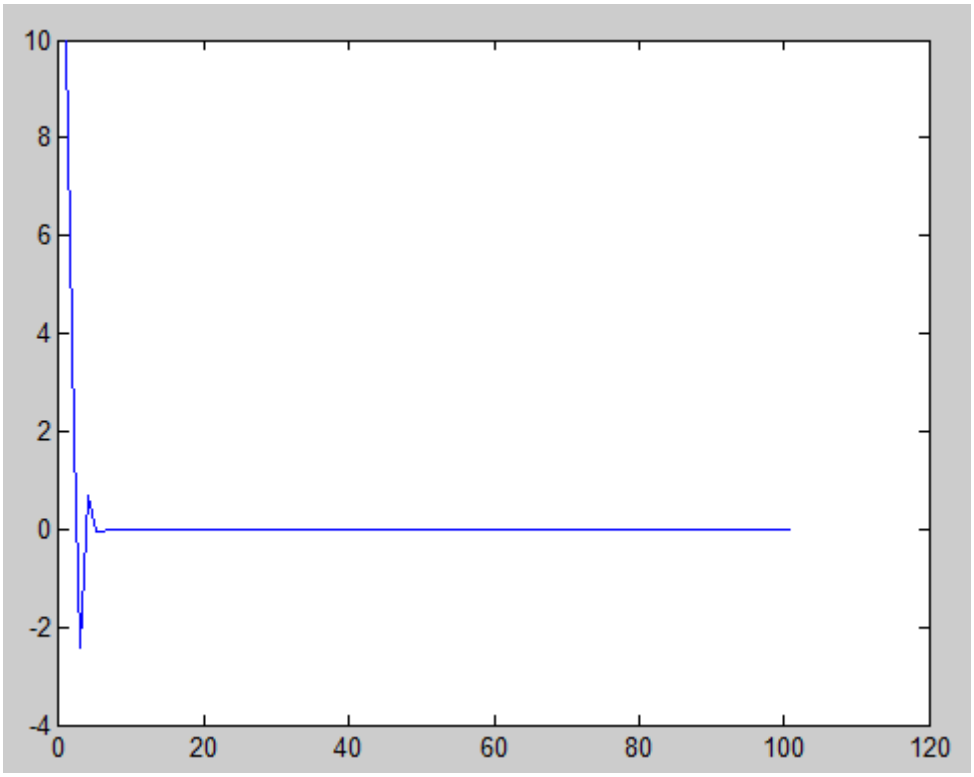
A kapott eredmény:



Futtatva a modellt az alábbi hibajelet kapjuk:

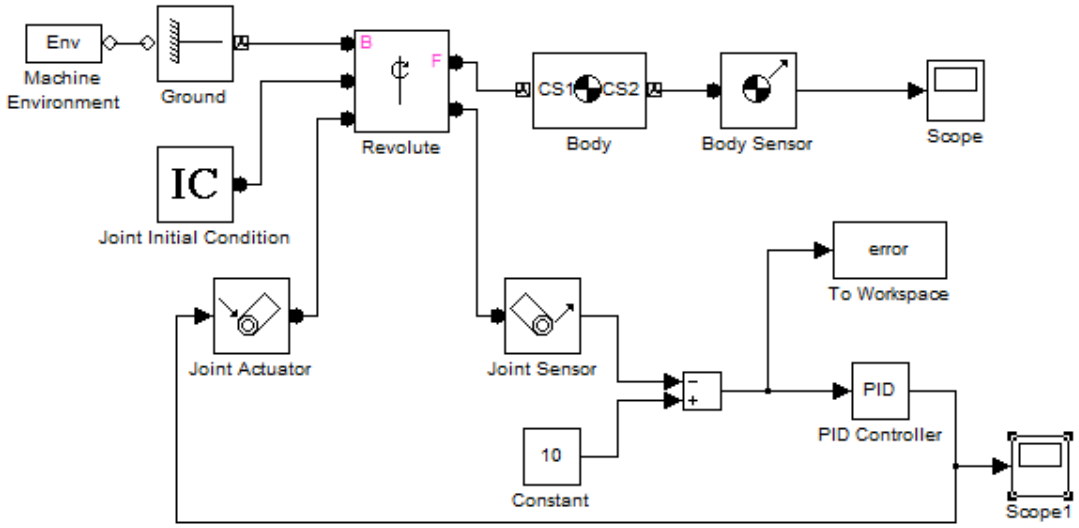
```
>> pend([5.564979054434392   3.362443131903184   0.31054545147032503])
```

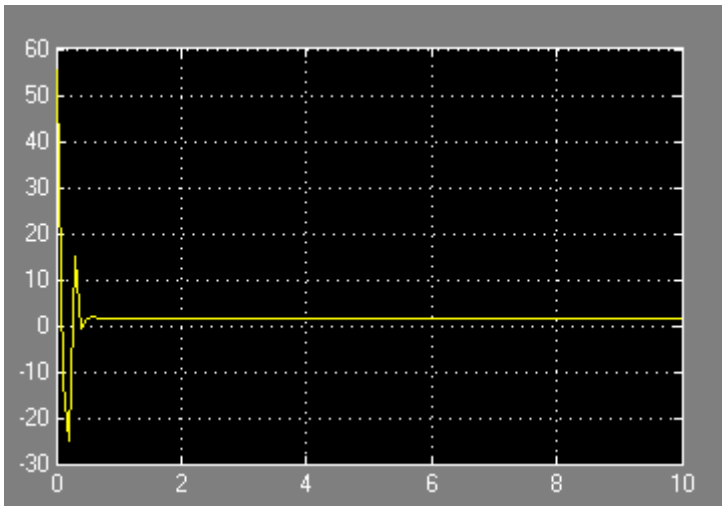
```
>> plot(error.signals.values)
```



Ahogy haladok a magasabb generációk felé egyre jobban nő a paraméterek értéke. Korlátozhatjuk az initial rang-et.

Nézzük meg mekkora nyomatékokat használ a rendszer.



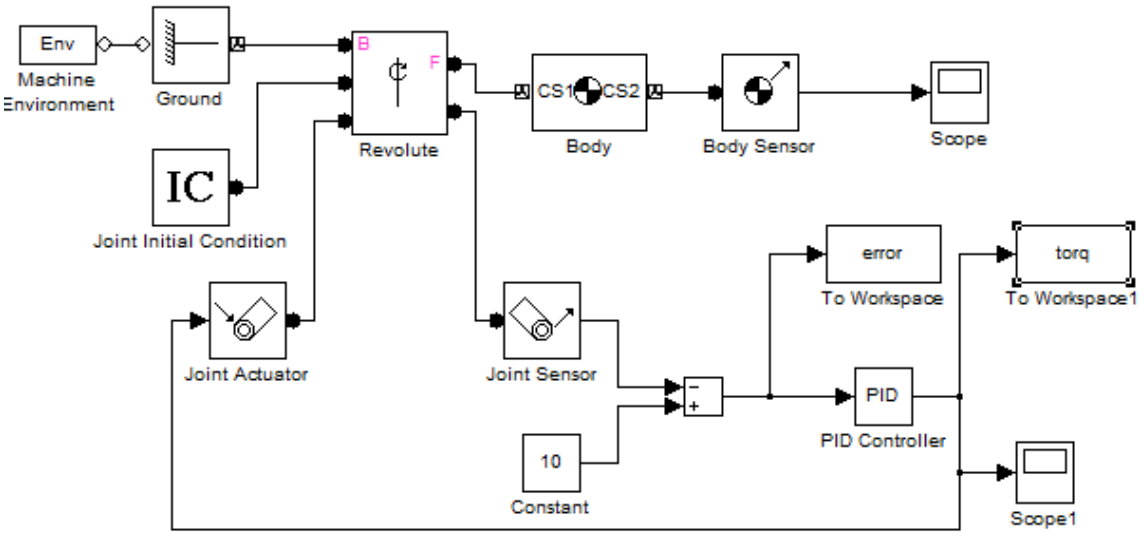


Nagyon nagyokat, próbáljuk meg ezt is belevenni az optimalizálásba.

Feladat:

1. Vegyük figyelembe a nyomatékokat is.
2. A nyomatékokat kisebb súllyal vegyük figyelembe.
3. Optimalizáljuk úgy a PID paramétereket hogy előbb 10 fokra majd 90 fokra térítjük ki a rendszert.

1. feladat megoldás:



```
function [ret] = pend(pars)
%input paraméterek workspace-be töltése
assignin('base', 'p_val', pars(1));
assignin('base', 'i_val', pars(2));
assignin('base', 'd_val', pars(3));

%modell definíció
mdl='pendopt.mdl';
%modell betöltése
load_system(mdl);
%modell szimulálása 0-tól 10 másodpercig a base workspace-ben
sim(mdl, 0:10, simset('DstWorkspace', 'base'));

%a szimuláció toworkspace doboza által visszaadott jelsorának
%kinyerése a base workspace-ből
err=evalin('base', 'error.signals.values');
torq=evalin('base', 'torq.signals.values');
%kinyert jelsor plottolása
%plot(y);
ret=sum(abs(y)+abs(torq));
end
```

2. feladat megoldás

```
function [ret] = pend(pars)
%input paraméterek workspace-be töltése
assignin('base', 'p_val', pars(1));
assignin('base', 'i_val', pars(2));
assignin('base', 'd_val', pars(3));

%modell definíció
mdl='pendopt.mdl';
%modell betöltése
load_system(mdl);
%modell szimulálása 0-tól 10 másodpercig a base workspace-ben
sim(mdl, 0:10, simset('DstWorkspace', 'base'));

%a szimuláció toworkspace doboza által visszaadott jelsorának
%kinyerése a base workspace-ből
err=evalin('base', 'error.signals.values');
torq=evalin('base', 'torq.signals.values');
%kinyert jelsor plottolása
%plot(y);
ret=sum(abs(y)+0.1*abs(torq));
end
```

3. feladat megoldás

```
%modell definíció
mdl='pendopt.mdl';
%modell betöltése
load_system(mdl);
%modell szimulálása 0-tól 10 másodpercig a base workspace-ben
sim(mdl, 0:10, simset('DstWorkspace', 'base'));
%a szimuláció toworkspace doboza által visszaadott jelsorának
%kinyerése a base workspace-ből
err10=evalin('base', 'error.signals.values');
torq10=evalin('base', 'torque.signals.values');
assignin('base', 'target_pos', 90);
load_system(mdl);
sim(mdl, 0:10, simset('DstWorkspace', 'base'));

err90=evalin('base', 'error.signals.values');
torq90=evalin('base', 'torque.signals.values');
%kinyert jelsor plottolása
%ret=sum(err);
%ret=sum(abs(err));
%ret = sum(abs(err)+abs(torq));
%ret = sum(abs(err)+0.1*abs(torq));
ret = sum(abs(err10)+0.3*abs(torq10)+abs(err90)+0.3*abs(torq90));
```