Departamentul Automatică și Informatică Industrială
Facultatea Automatică și Calculatoare
Universitatea POLITEHNICA din București

Proiect SCPI
Reglarea presiunii

Grupa 341 A2
Ioana-Roxana Boriceanu
Andreea-Gabriela Nițoi

2023

Cuprins

Prezentarea Platformei **3**
Schema instalației 3
Componente 4

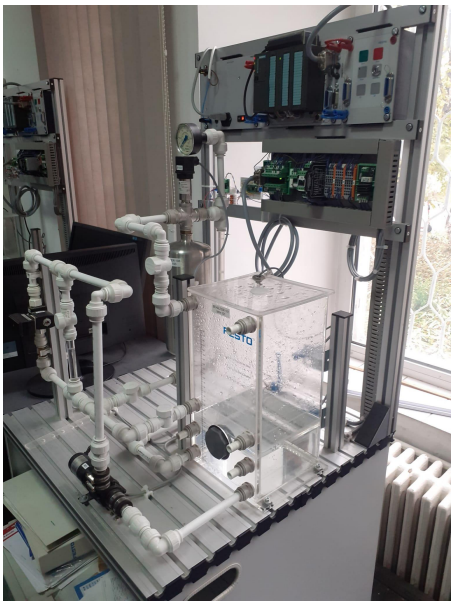
Identificare experimentală **5**

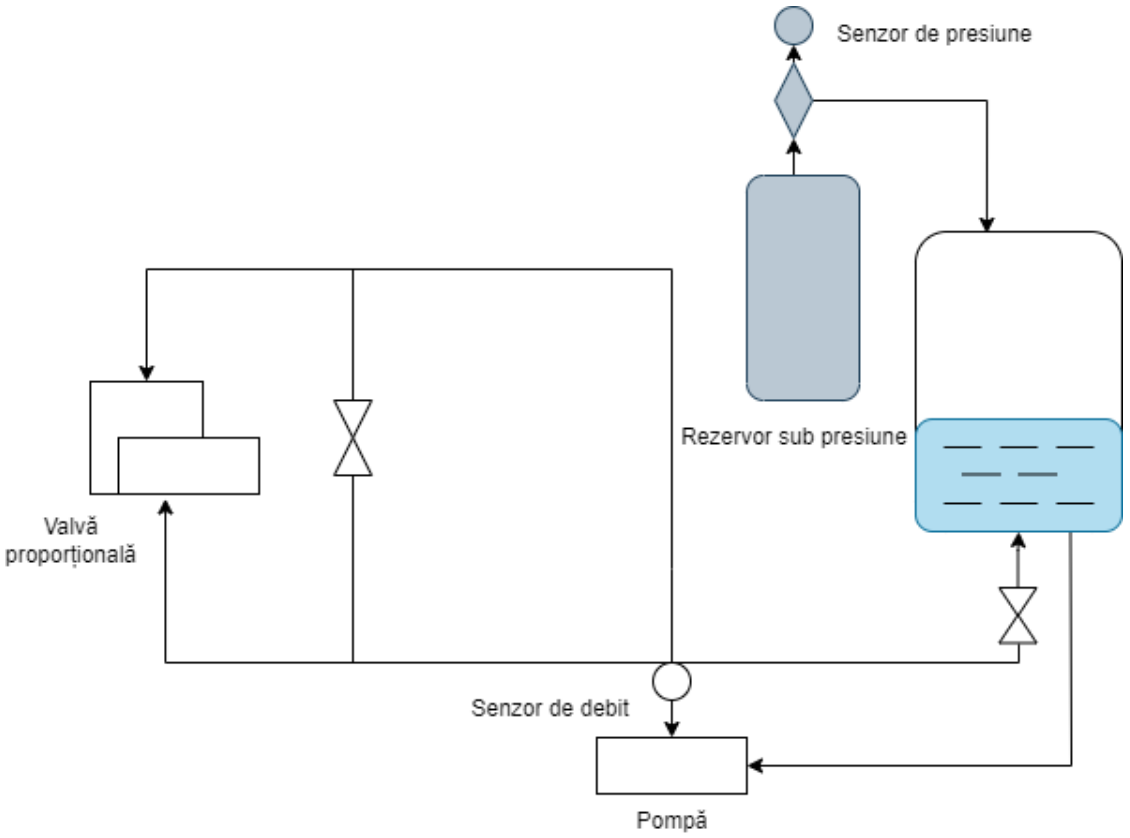
Regulator PID **8**
Legea de comandă 11
Validarea pe platformă 12

Regulator RST **14**
Validarea pe platformă 15

Prezentarea Platformei

Schema instalației

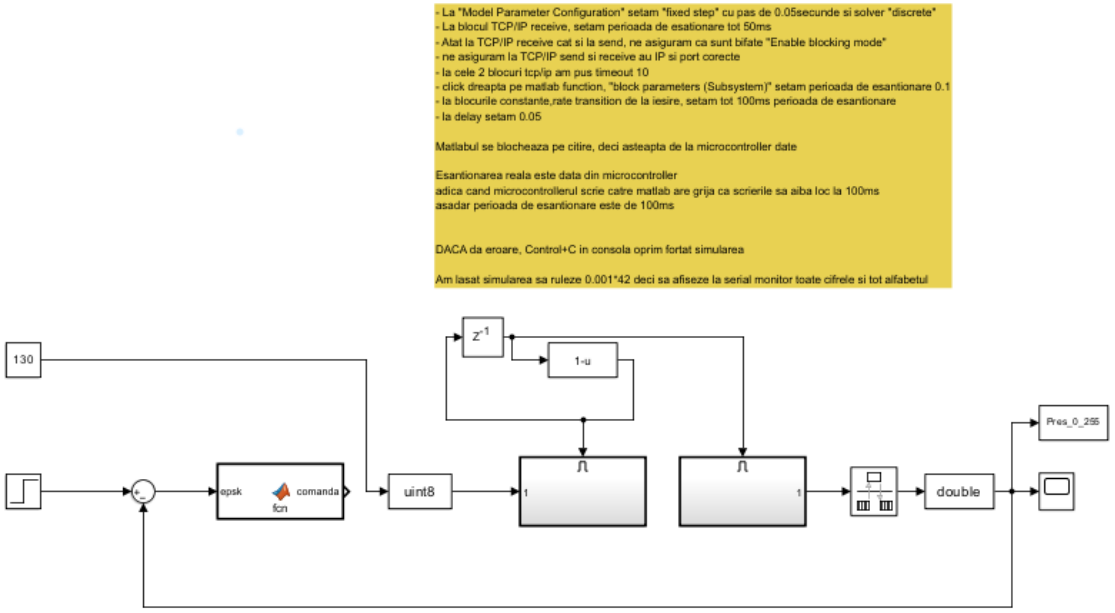




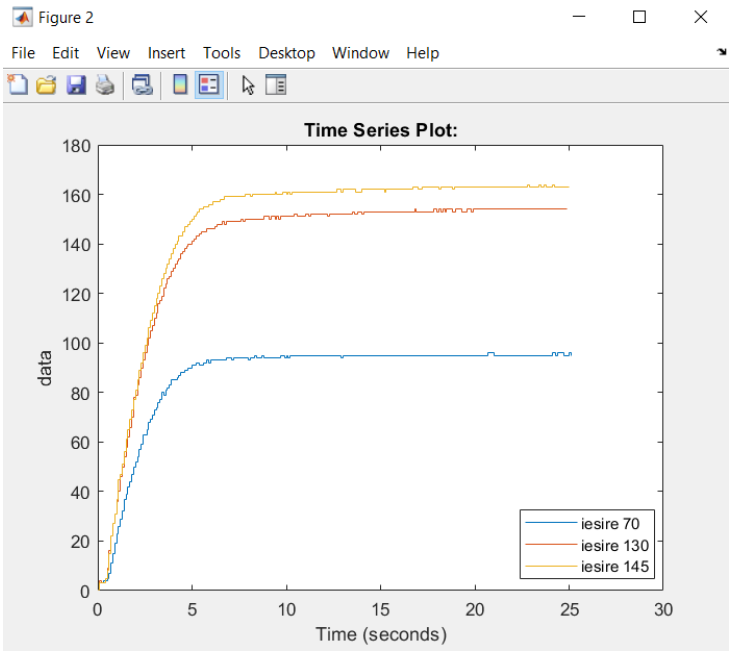
Componente

1. **Manometru** - masoară presiunea și o afișează pe ceas. Presiunea maximă pe care o poate măsura este de ¾ din valoarea maximă afișată, adică 8 bar.
2. **Senzor de presiune** - convertește semnalul măsurat (presiunea) în semnale de ieșire a căror valoare este aflată într-unul din intervalele: 4…20mA, 0…20mA sau 0…10V.
3. **Pompă** - se atașează printr-o clemă inelarp și este montată utilizând două șuruburi și piulițe cu cap T. Are o putere de 26W, iar tensiunea de alimentare este 24V.
4. **Supapă proporțională** - controlează debitul lichidului care trece prin ea. Cursa pistonului supapei este dată de tensiunea bobinei pe care o conține, echivalentă cu valoarea debitului. Semnalul de intrare poate avea o valoare aflată într-unul din următoarele intervale: 0…10V, 0…20mA, 4…20mA.

Identificare experimentală



Pentru achiziția de date, am citit răspunsurile la comanda de tip treaptă din trei simulări diferite. Datele obținute le-am salvat în trei variabile de tip timeseries pe care mai apoi le-am prelucrat și le-am salvat împreună în fișierul „date.mat”.
Răspunsurile din cele trei simulări se pot observa și în figura de mai jos:



Pentru fiecare ieșire am construit câte o funcție de transfer care aproximează răspunsul la treaptă al procesului.

```
%% comanda 70
s = tf('s');
Hs = (1.35/(1.42*s + 1)) * exp(-0.5*s);

figure
plot(iesire_70)
hold on
step(70*Hs)
```

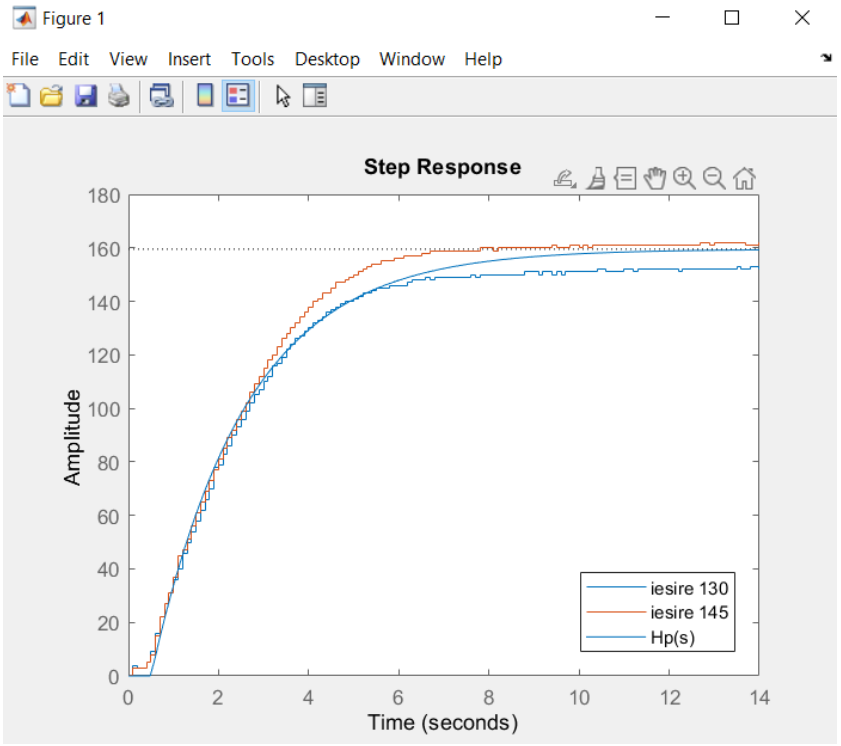
```
%% comanda 130

Hs = (1.17/(1.80*s + 1)) * exp(-0.5*s);

figure
plot(iesire_130)
hold on
step(130*Hs)
```

```
%% comanda 145

Hs = (1.10/(1.74*s + 1)) * exp(-0.5*s);

figure
plot(iesire_145)
hold on
step(145*Hs)
```

Apoi din cele trei funcții am obținut funcția finală pe care am folosit-o pe tot parcursul simulării, de forma $Hs = (1.16/(2.1 * s + 1)) * exp(-0.5 * s)$.

```
%% Functia de transfer a procesului

Hs = (1.16/(2.1*s + 1)) * exp(-0.5*s);

figure
plot(iesire_130)
hold on;
plot(iesire_145)

hold on
step(137.5*Hs)
hold off;
```

Deoarece nu am reușit să găsim o funcție care să aproximeze foarte bine toate cele trei răspunsuri citite, am renunțat la comanda 70 atunci când am creat funcția de transfer a procesului.



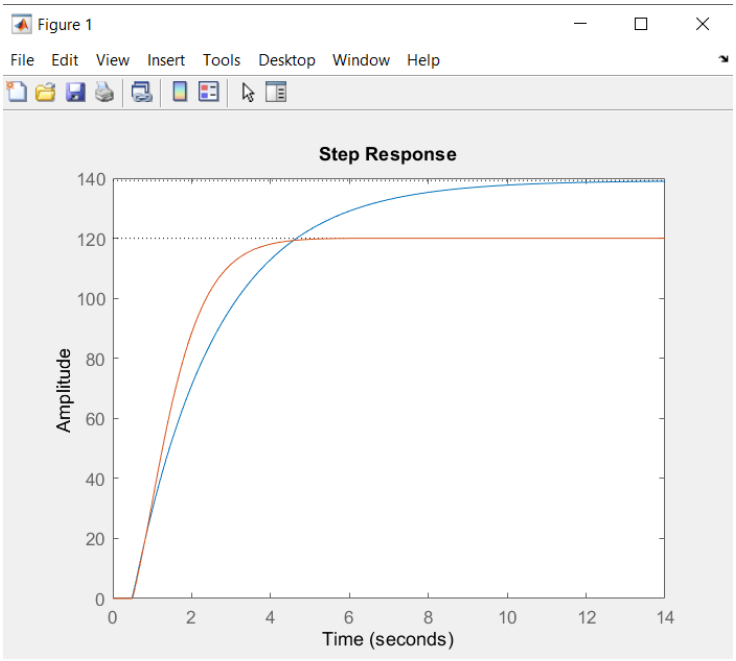
Se poate observa că cea mai bună aproximare am obținut-o pentru o funcție de transfer de ordinul I, cu timp mort de 0.5s.

Regulator PID

Deoarece funcția noastră de transfer are timp mort, am proiectat un regulator PI cu predictor Smith.

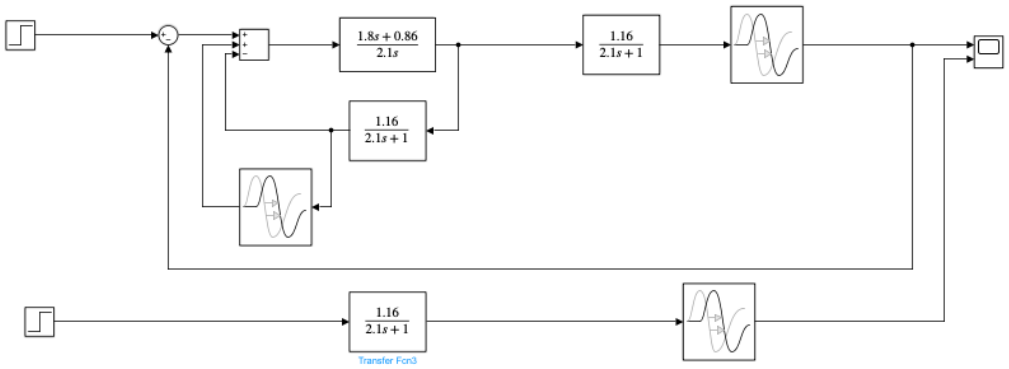
```
%% Regulator PI cu metoda predictor Smith
Hs = (1.16/(2.1*s + 1)) * exp(-0.5*s);

Hr = (0.86*(2.1*s+1))/(2.1*s);
Hr_disc=c2d(Hr,0.1);

% pentru verificare
Hd=(1.8*s+0.86)/(2.1*s-1+exp(-0.5*s));

figure
step(120*Hs)
hold on
step(120*(feedback(Hd*Hs,1)))
hold off

Hd_disc=tf(c2d(Hd,0.1));
```

Pentru verificarea robusteții regulatorului, am creat o altă schemă în Simulink ce conține atât procesul și regulatorul în formă continuă, cât și cele două componente în formă discretizată. Funcția Hd, folosită pentru verificarea corectitudinii regulatorului PI după discretizare a fost obținută prin calculul regulatorului echivalent pentru forma cu predictor Smith.

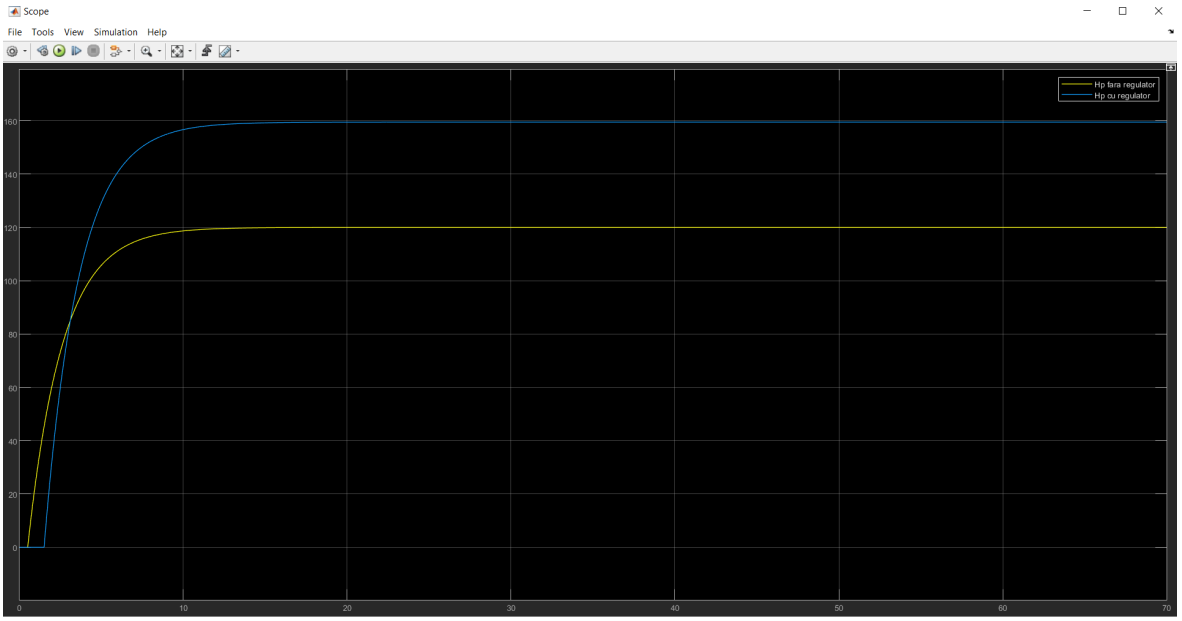


În figura de mai sus se poate observa cu albastru funcția de transfer a procesului, iar cu roșu răspunsul procesului la comanda 120 după introducerea regulatorului.

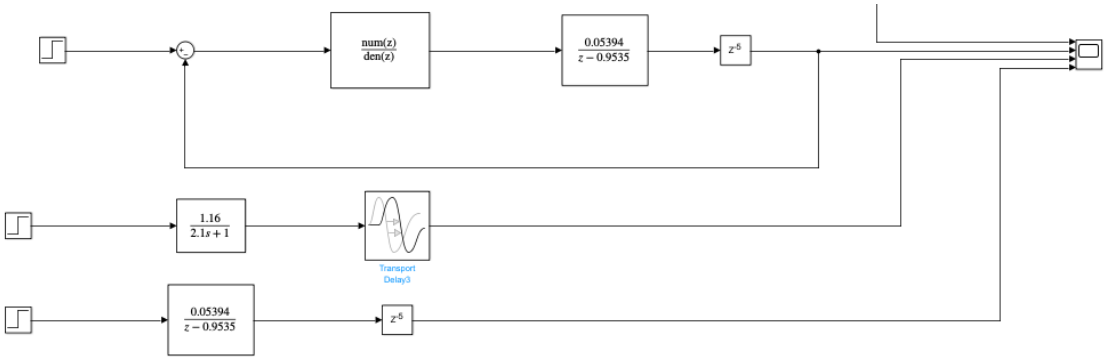
Pentru verificarea regulatorului în regim continuu, am folosit următoarea schemă Simulink.



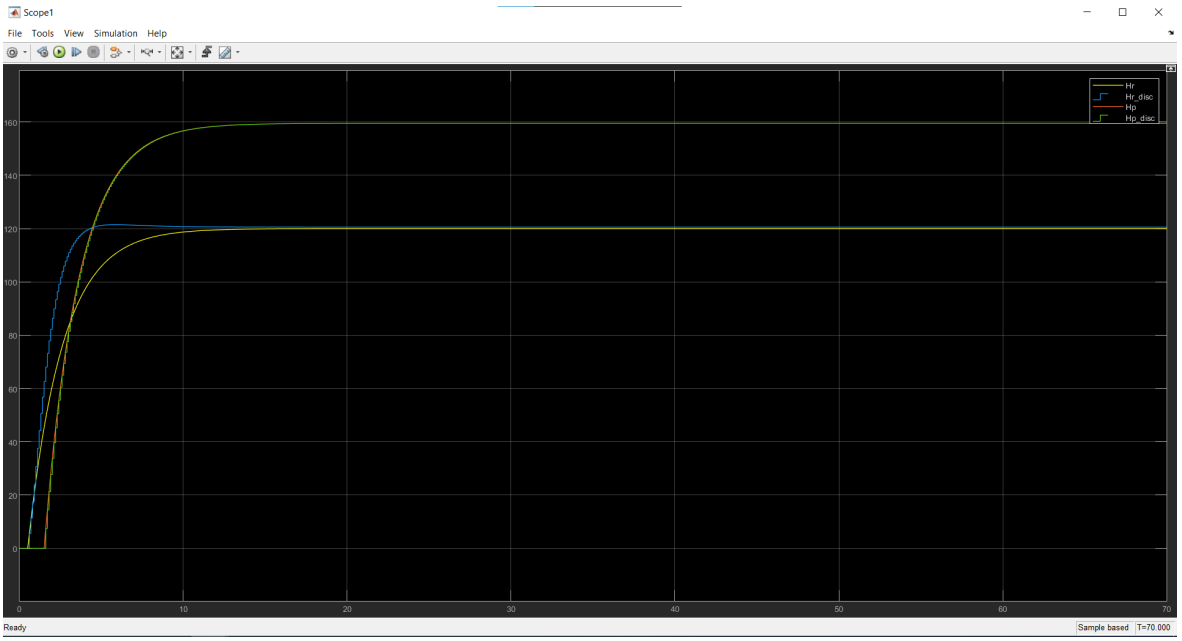
Mai jos se pot observa răspunsurile procesului în continuu pentru comanda 120, cu și fără regulator.



Pentru verificarea regulatorului în regim discret, am folosit următoarea schemă Simulink.



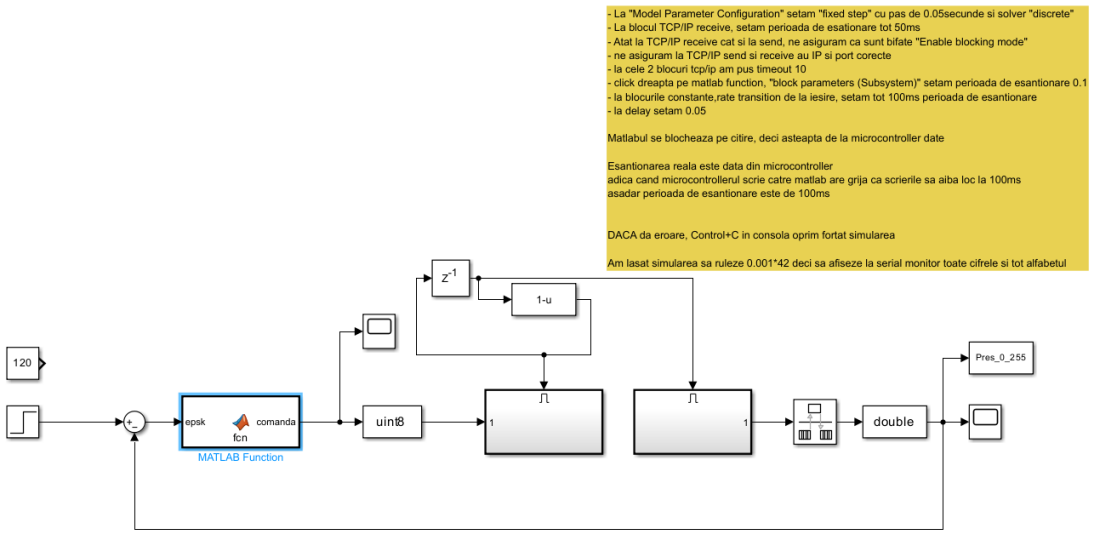
Mai jos se pot observa răspunsurile procesului atât în continuu, cât și în discret pentru comanda 120, cu și fără regulatoare pentru ambele forme.



Legea de comandă

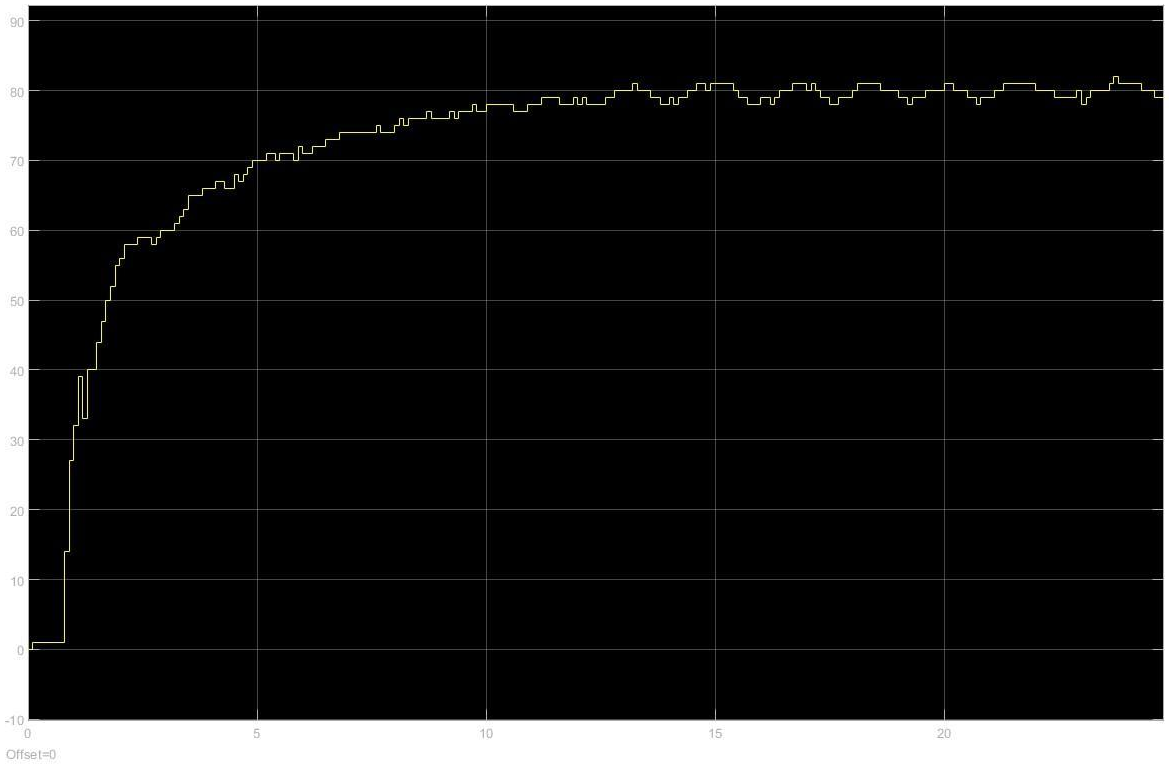
Pentru a putea testa regulatorul pe platformă am folosit ecuatia Hr_disc[z]=u[k]/e[k] și am obținut expresia:
*uk=0.86*epsk-0.819*epsk_1+uk_1;*
Pe care am introdus-o în schema Simulink a instalației în blocul Matlab function.

```
function comanda = fcn(epsk)
persistent uk_1 epsk_1
uk=0;
if isempty(uk_1)
    uk_1=0;
    epsk_1=0;
else
uk=0.86*epsk-0.819*epsk_1+uk_1;
uk_1 = uk;
epsk_1 = epsk;
end

comanda=uk;
```

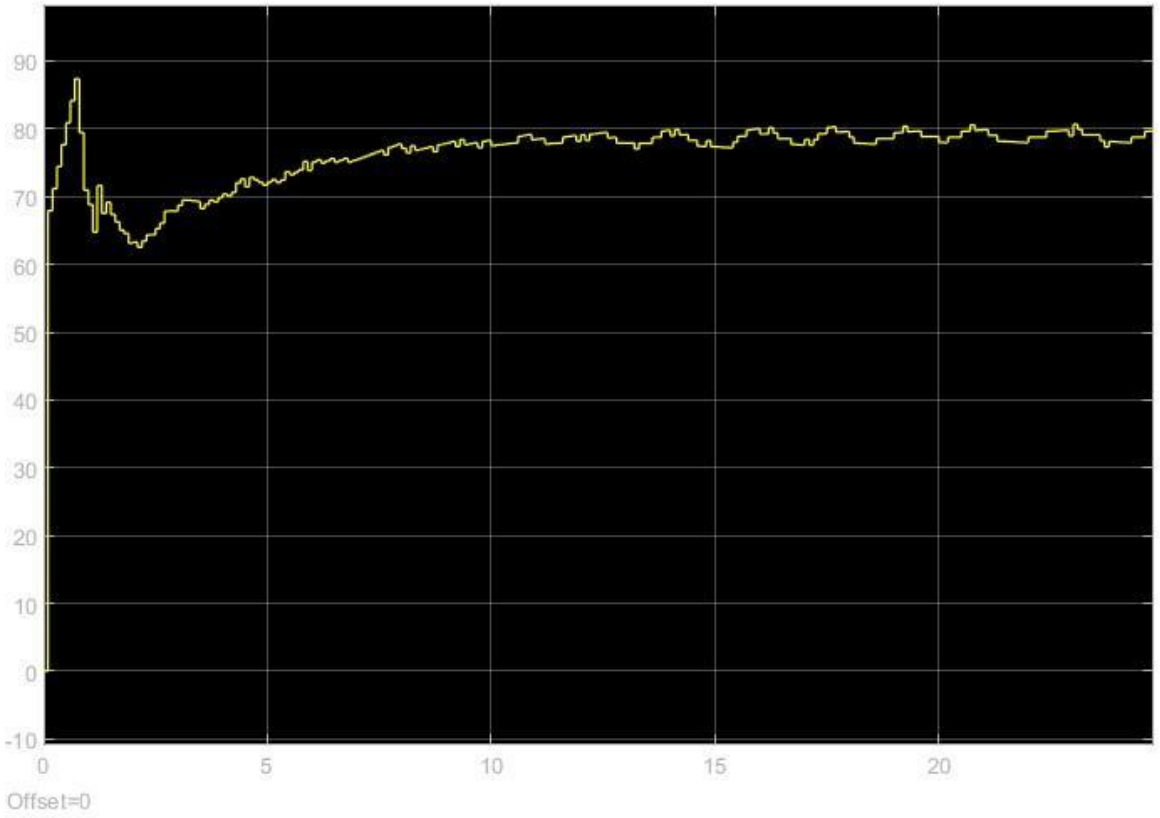


Validarea pe platformă

În urma simulării pe platformă am obținut următorul raspuns pentru comanda 80.

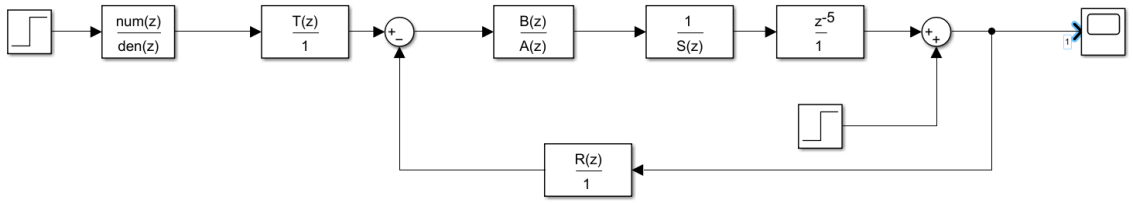


Graficul comenzii este cel de mai jos:

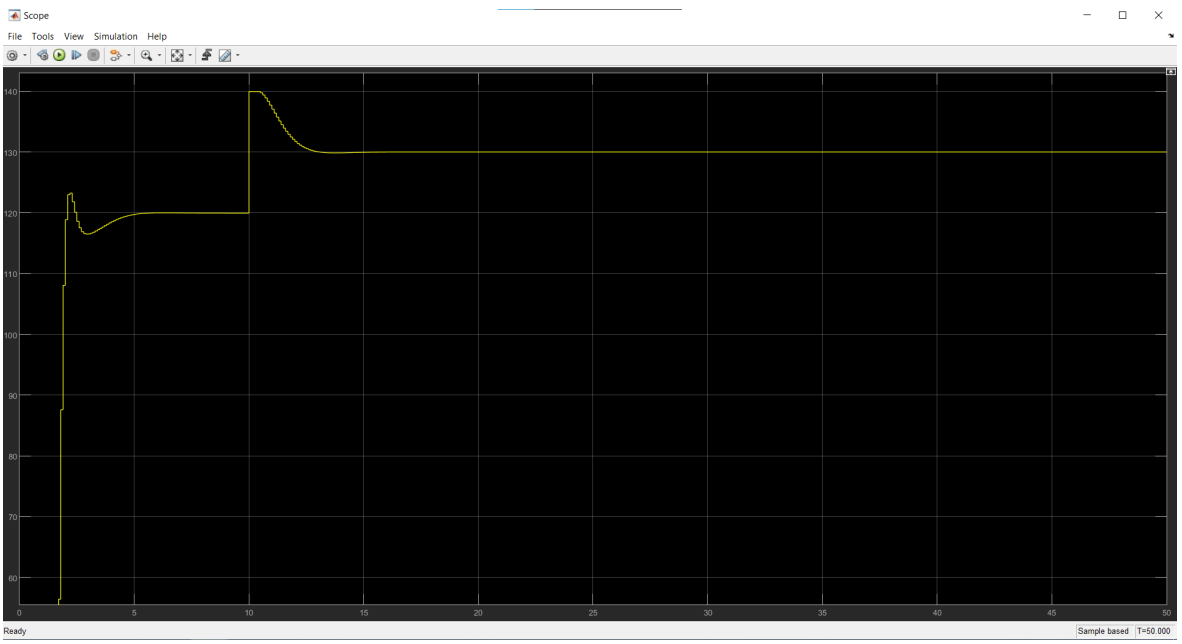


Regulator RST

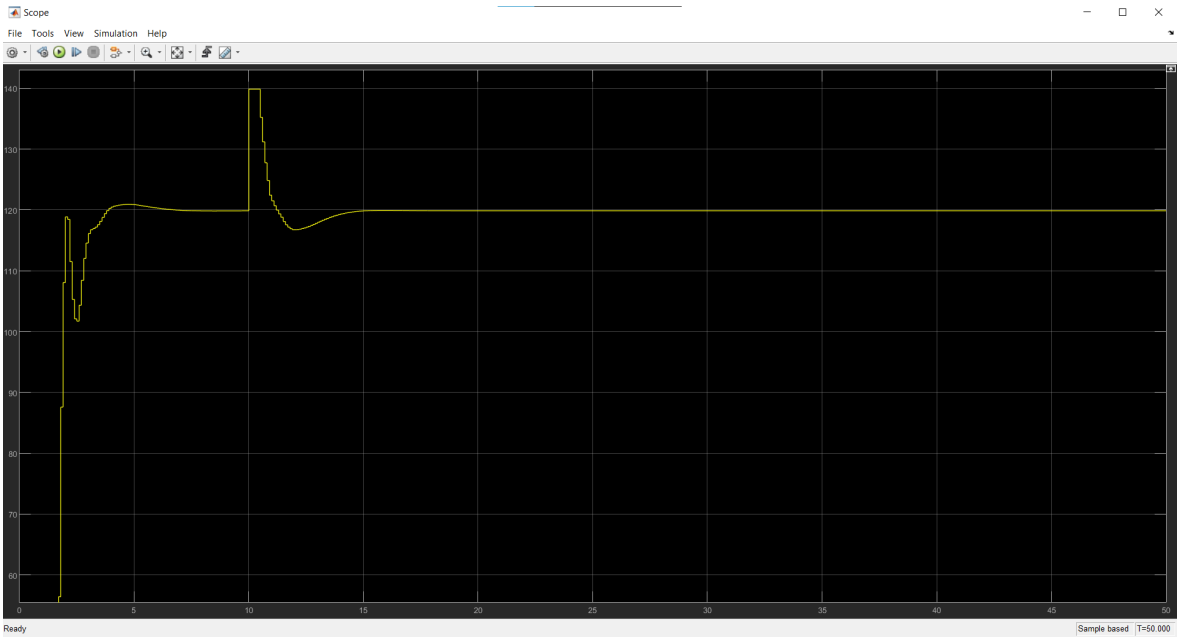
Pentru regulatorul RST am implementat în Matlab două forme: prima RST simplu, fără integrator forțat, a doua RST cu integrator forțat. Pentru verificarea acestuia, am creat o nouă schemă Simulink ca în figura de mai jos.



Primul bloc din schemă se numește generator de traiectorie și l-am obținut din funcția de transfer H_impus, la care am modificat valoarea pulsației înmulțind-o cu timpul tranzitoriu, care în cazul nostru are valoarea 4.6.
Pentru regulatorul fără integrator forțat am obținut următorul răspuns pentru comanda 120:

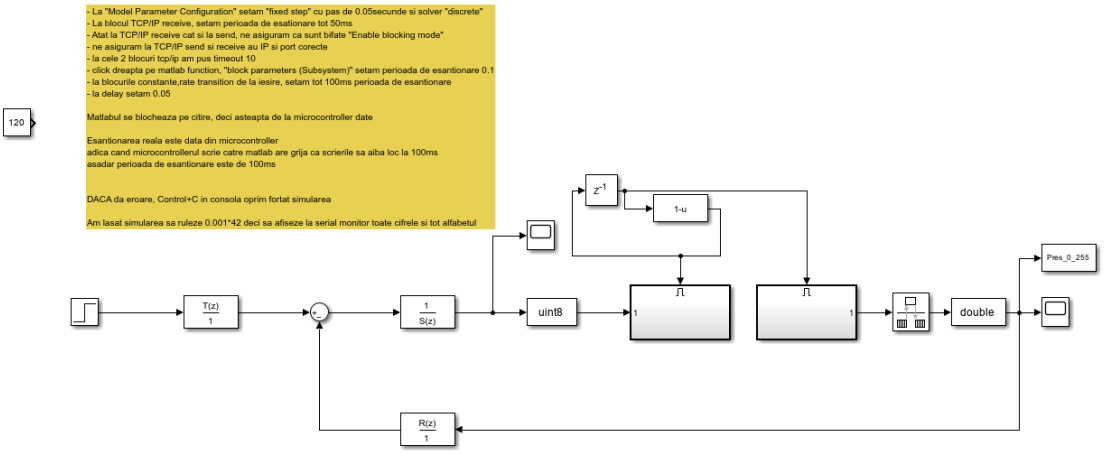


Se poate observa pe grafic faptul că nu este urmărită referința. Pentru a rezolva această problemă, am creat cel de-al doilea regulator, pentru care am obținut, pentru aceeași comandă, următorul răspuns:

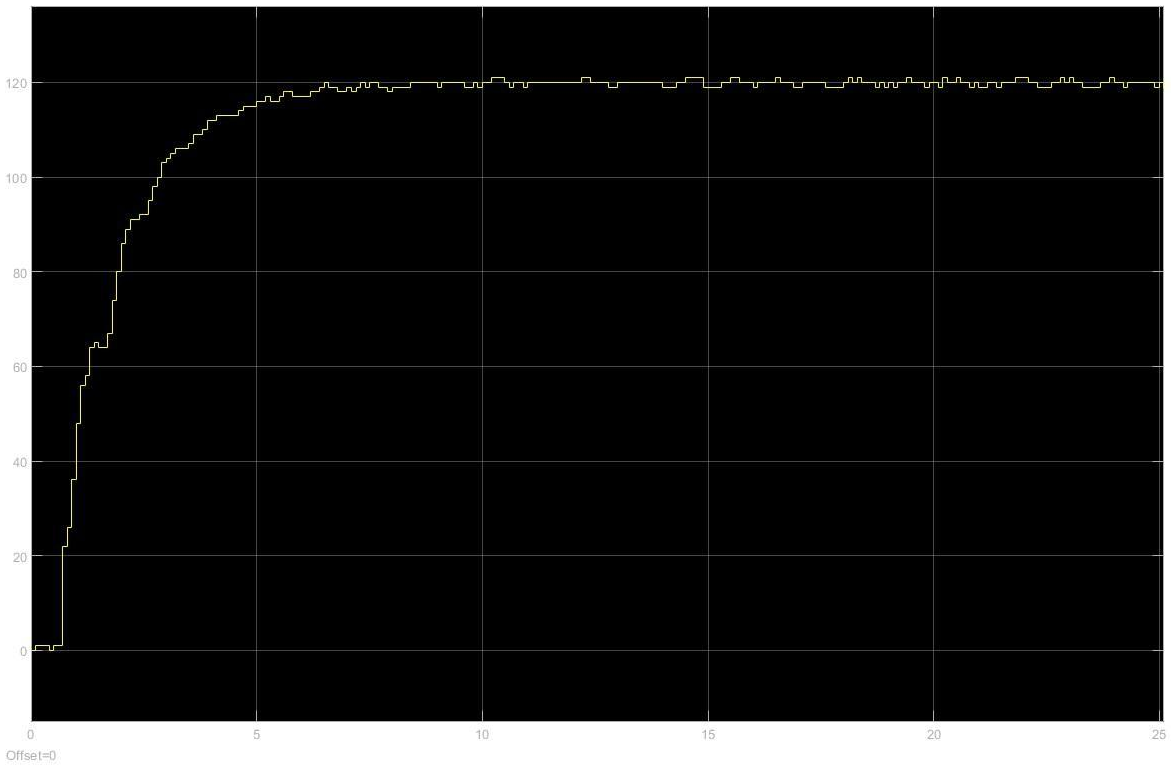


Validarea pe platformă

Pentru validarea pe platformă am utilizat RST cu integrator forțat.



În urma simulării, pentru comanda 120, am obținut următorul răspuns.



Graficul comenzii: