Modelare si Simulare
Tema laborator
-
Tema 2
Instalatie hidraulica cu patru rezervoare

Ionescu Alexandru Cristian
Pangratie Andrei
333 AC

2018
Decembrie

Introducere

Simularea procesului este salvata cu versiunea MATLAB R2017a si pentru rularea ei se va folosi scriptul *run_tema_2017.m*. Pentru a rula cu MATLAB R2018a se va folosi scriptul *run_tema.m*

Structura proiectului

Ficare subpunct are doua fisiere aferente (*load_workspace_*.m* si *tema_comm_*.m*)
Pentru a rula simulari individuale se pot comenta/decomenta doua cate doua liniile respective.
Fisierul *animate_levels.m* este folosit dupa fiecare simulare pentru a crea o reprezentare grafica a evolutiei nivelelor rezervoarelor cu apa.
Pentru simularile repetate, animatia se poate dezactiva prin setarea flagului: $animation\_enable = 0$

Subpunctul a

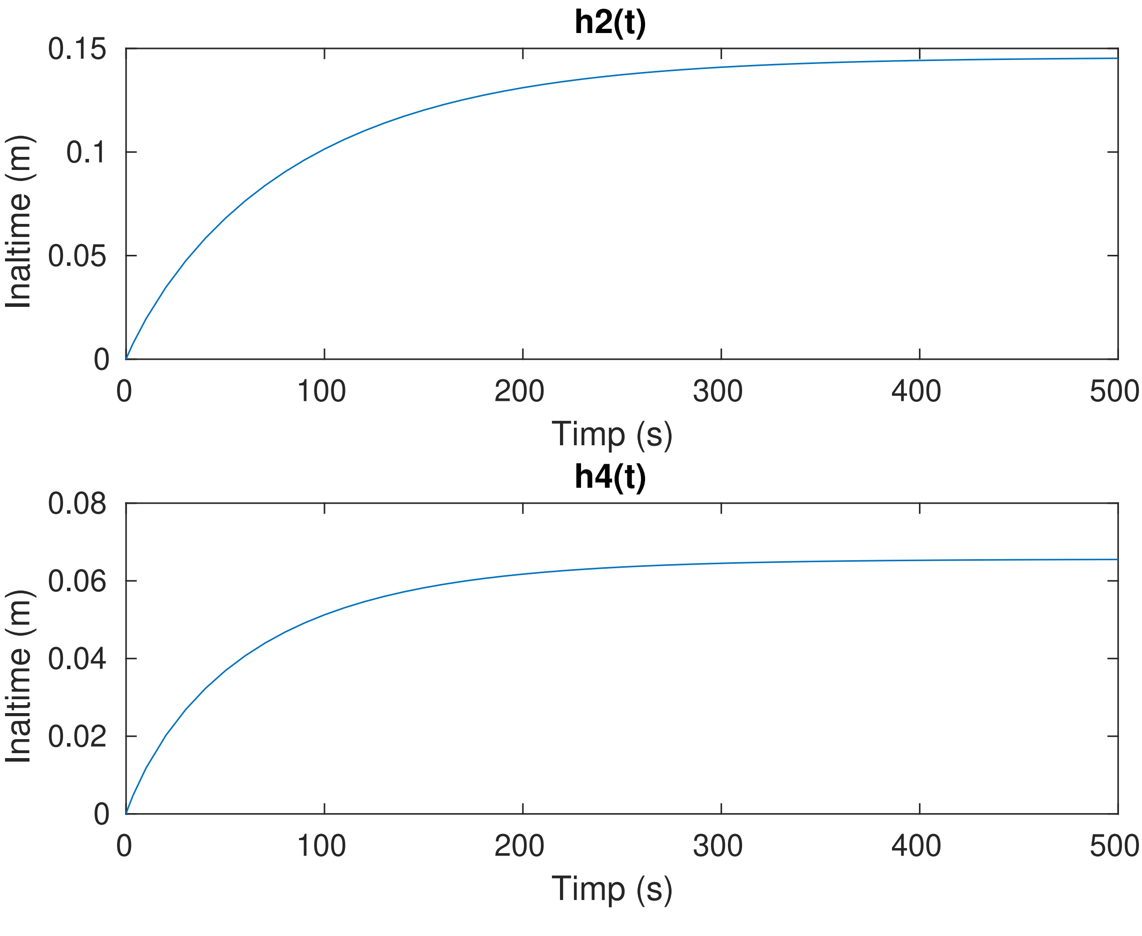

Figure 1: Evolutia iesirilor y2 si y4

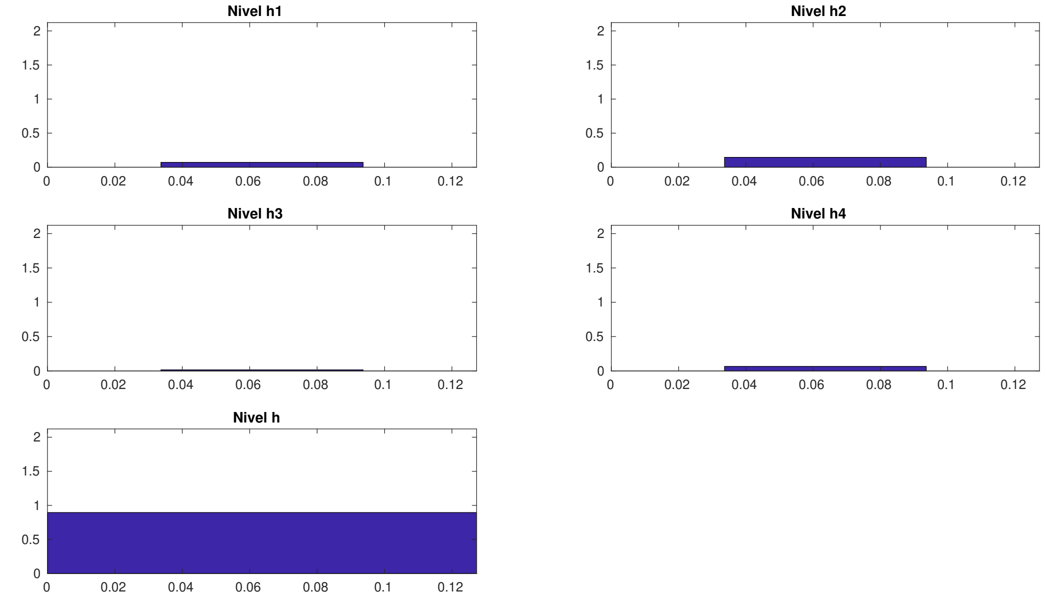

Figure 2: Exemplu animate levels

Subpunctul b

Pentru a varia conditiile initiale am creat vectorii *arr_cih, arr_cih1, arr_cih2, arr_cih3, arr_cih4* in fisierul *load_workspace_b.m*, iar pentru a rula mai multe simulari se pot adauga dupa nevoie valori noi acestora.

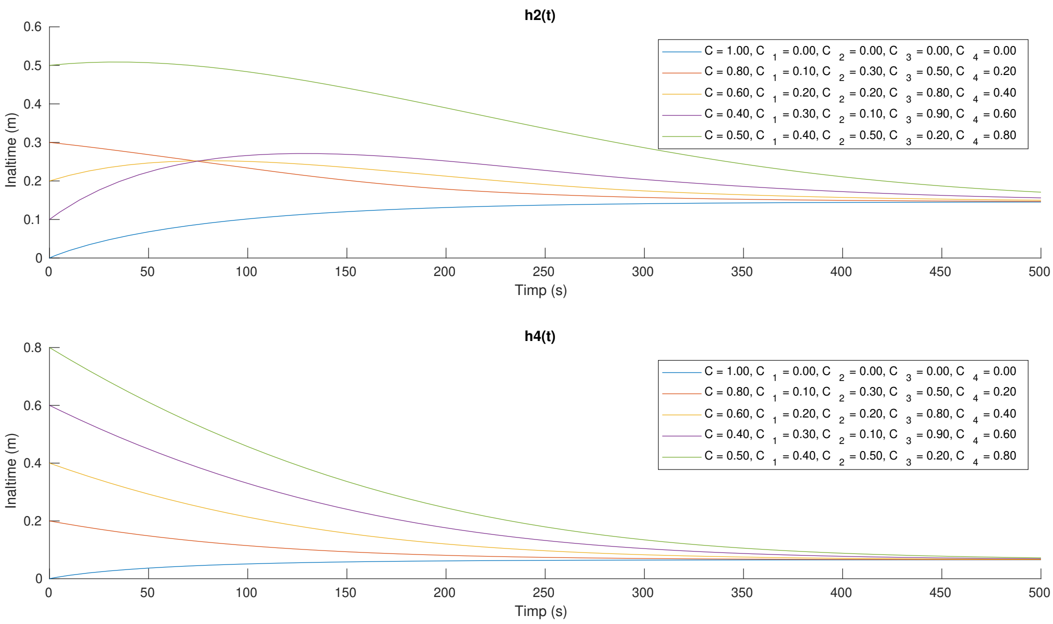

Figure 3: Evolutia iesirilor y2 si y4

Se observa ca indiferent de volumul initial de lichid din rezervoare, iesirile sistemului se vor stabiliza in jurul aceluias punct.

Subpunctul c

Folosing functia *stepinfo(h2.Data, h2.Time)* aflam informatii despre raspunsul sistemului si putem extrage timpul de stabilizare.

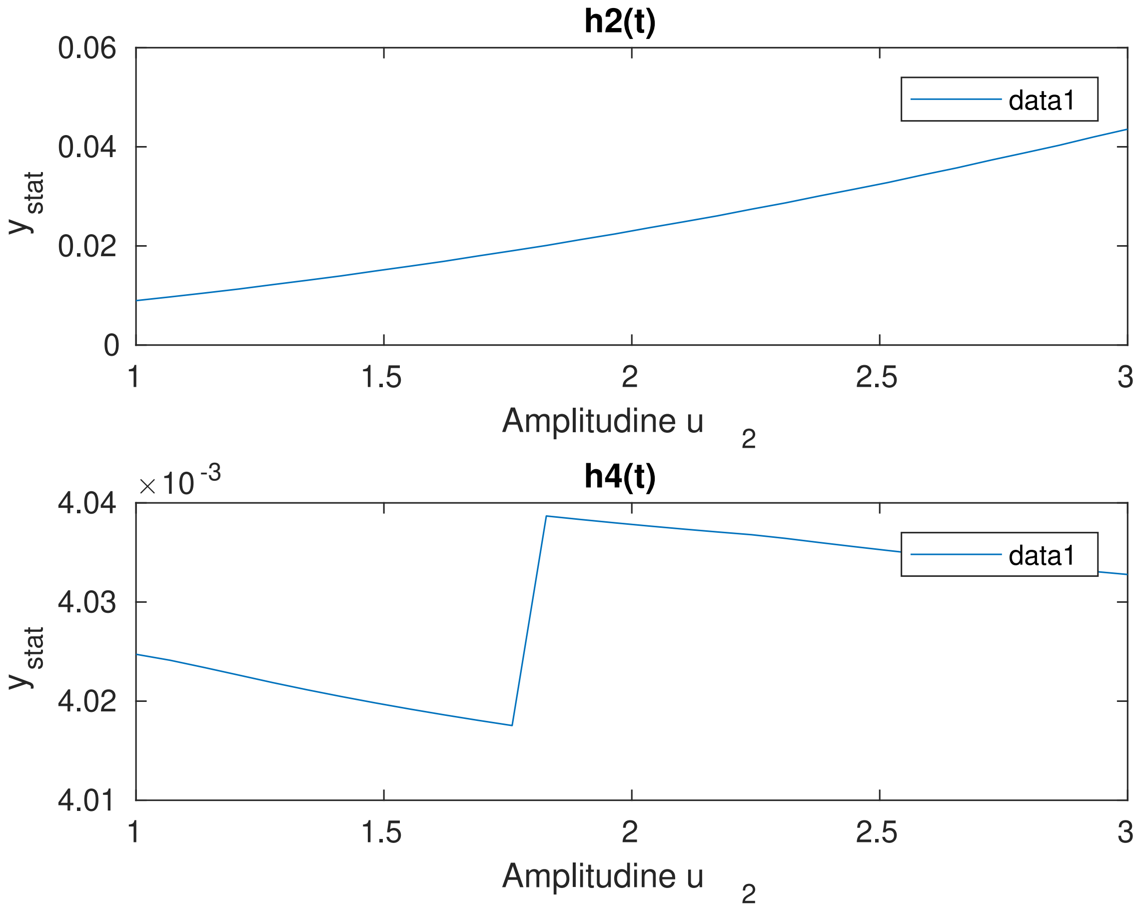

Figure 4: Evolutia iesirilor y2 si y4 in functie de valorile constantelor de integrare

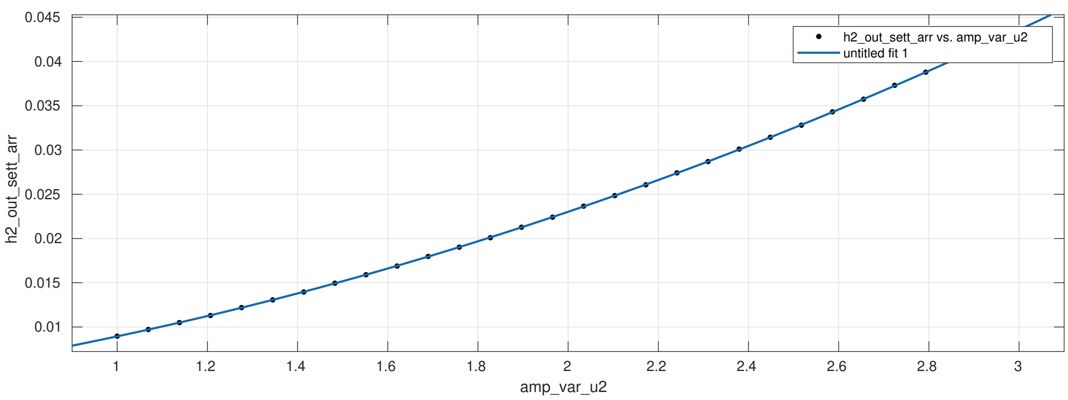

Figure 5: Stabilizare h2 in functie de amplitudinea intrarii

Cea mai buna aproximare se obtine cu polinomul de gradul 2:

$$f(x) = p1 * x^2 + p2 * x + p3, \begin{cases} p1 = 0.003242 \\ p2 = 0.004348 \\ p3 = 0.001353 \end{cases}$$

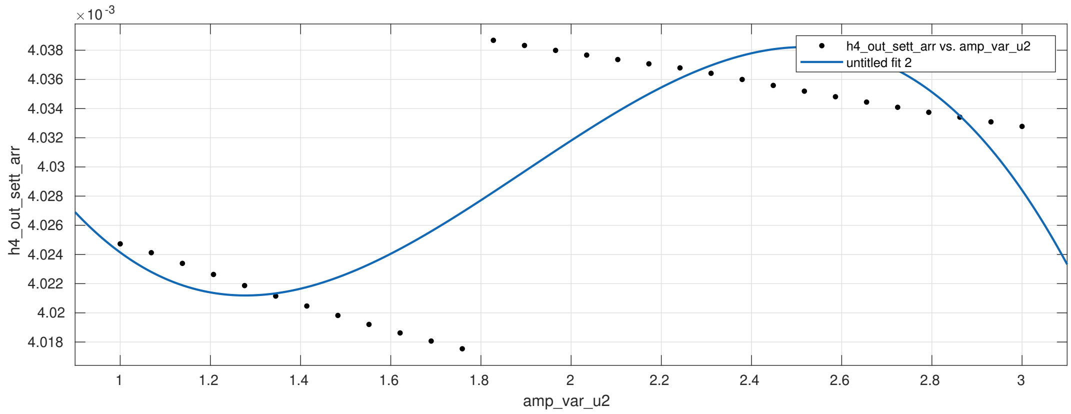

Figure 6: Stabilizare h4 in functie de amplitudinea intrarii

$$f(x) = p1 * x^3 + p2 * x^2 + p3 * x + p4, \begin{cases} p1 = -1.791e-05 \\ p2 = 0.000101 \\ p3 = -0.000172 \\ p4 = 0.004113 \end{cases}$$

Subpunctul d

Dupa trecerea timpului de stabilizare am aplicat urmatorul algoritm pentru a afla aproximarea liniara, unde j este primul index dupa stabilizare:

```
[A, B, C, D] = linmod(model_name, ...
        [h.Data(j) h1.Data(j) h2.Data(j) h3.Data(j) h4.Data(j)], ...
        [u1_data(j) u2_data(j)]);

linsys = ss(A, B, C, D);
out = lsim(linsys, [u1_data; u2_data]', u1_time');

ti = linspace(0, 300, 300);

nonlinear_out_2 = interp1(h2.Time, h2.Data, ti);
linear_out_2 = interp1(u2_time, out(:, 3), ti);
```

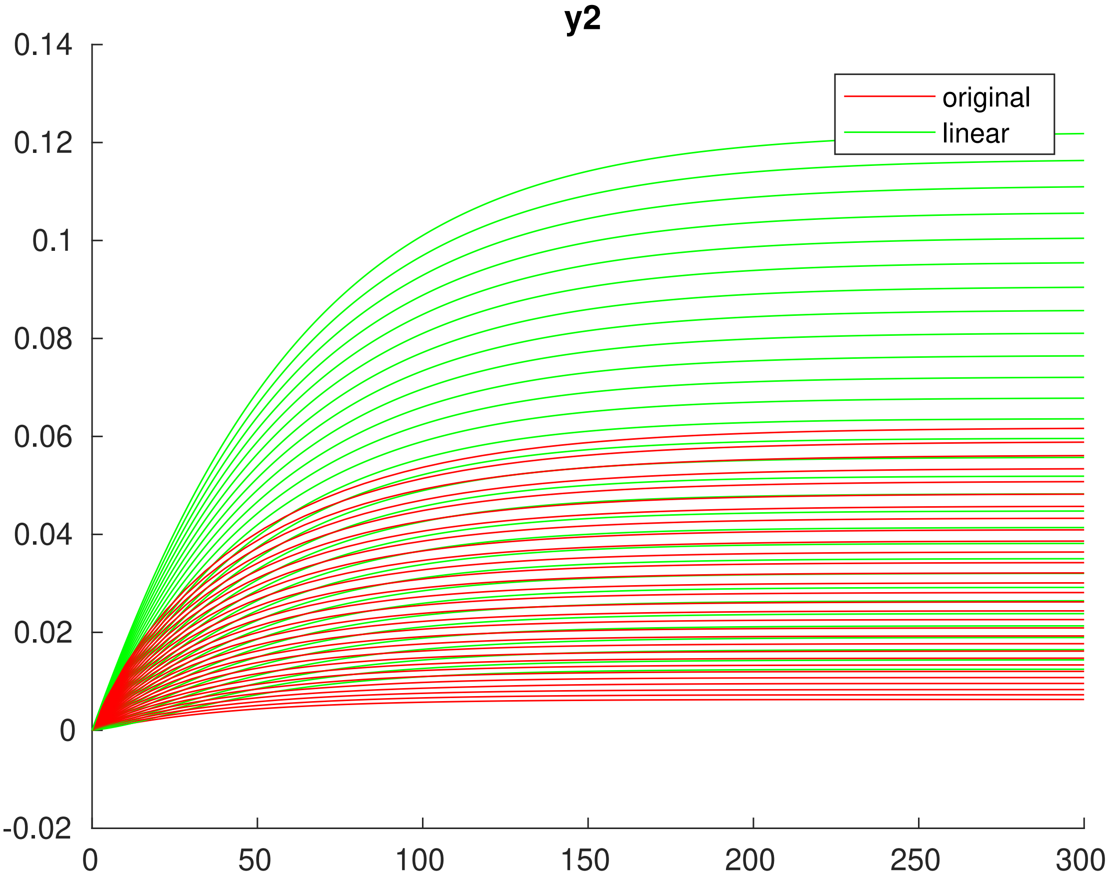

Figure 7: Evolutia iesirii h2 vs aproximarea sa liniara

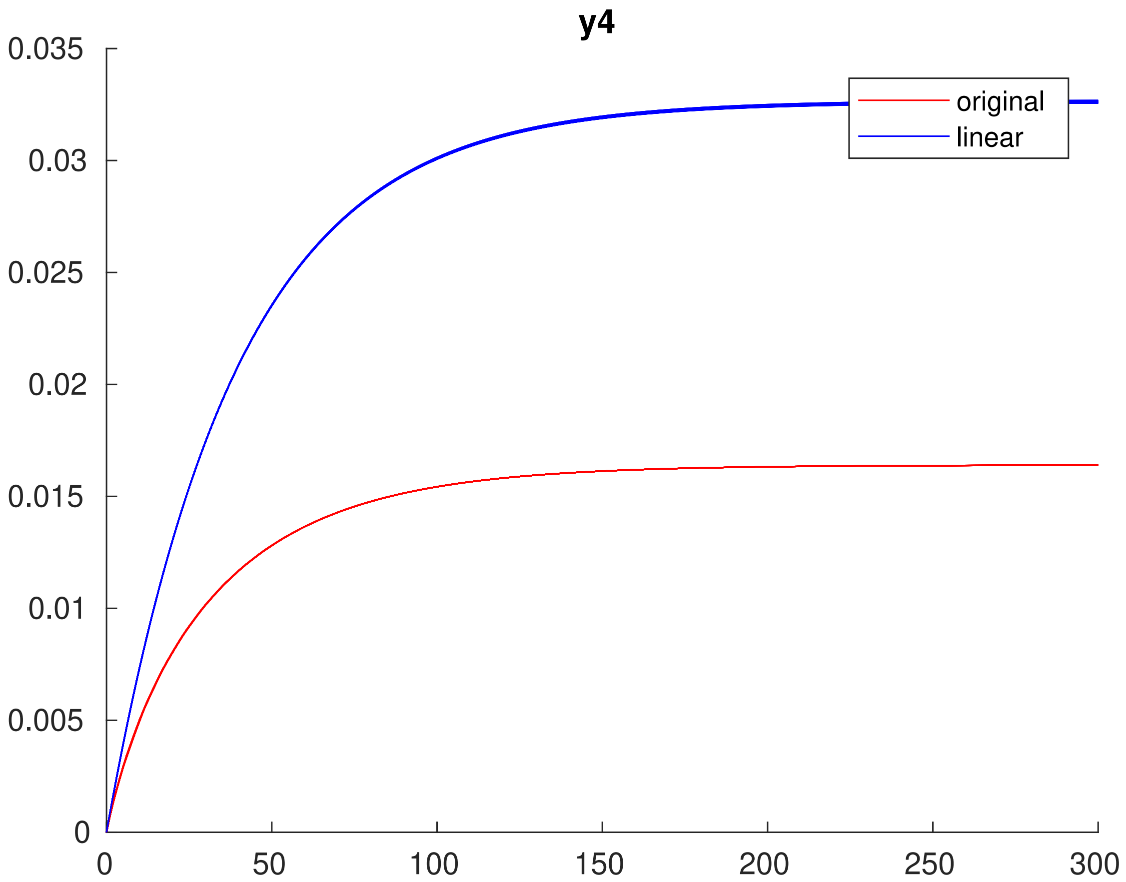

Figure 8: Evolutia iesirii h4 vs aproximarea sa liniara

Subpunctul e

Pentru a putea calcula eroarea cu ajutorul formulei din enunt, am interpolat vectorii iesirilor simiularii si aproximarii sale liniare astfel, stocand rezultatul in variabilele *error_sys_2(i, j)* si *error_sys_4(i, j)*:

```
linsys = ss(A, B, C, D);

for j=1:loop_len
        amp_u2 = amp_var_u2(j);

        u2_time = 0:time_step:time_count;
        u2_data = ones(1, length(u2_time)) * amp_u2;
        u2 = [u2_time' u2_data'];

        res = sim(model_name,'StartTime','0','StopTime','300','FixedStep','0.02');

        h  = res.h;
        h1 = res.h1;
        h2 = res.h2;
        h3 = res.h3;
```

```
    h4 = res.h4;

    out = lsim(linsys, [u1_data; u2_data]', u1_time');

    ti = linspace(1, 300, 300);

    nonlinear_out_2 = interp1(h2.Time, h2.Data, ti);
    linear_out_2 = interp1(u2_time, out(:, 3), ti);

    nonlinear_out_4 = interp1(h4.Time, h4.Data, ti);
    linear_out_4 = interp1(u2_time, out(:, 5), ti);

    error_sys_2(i, j) = ...
            norm((linear_out_2 - nonlinear_out_2) ./ nonlinear_out_2);

    error_sys_4(i, j) = ...
            norm((linear_out_4 - nonlinear_out_4) ./ nonlinear_out_4);
end
```

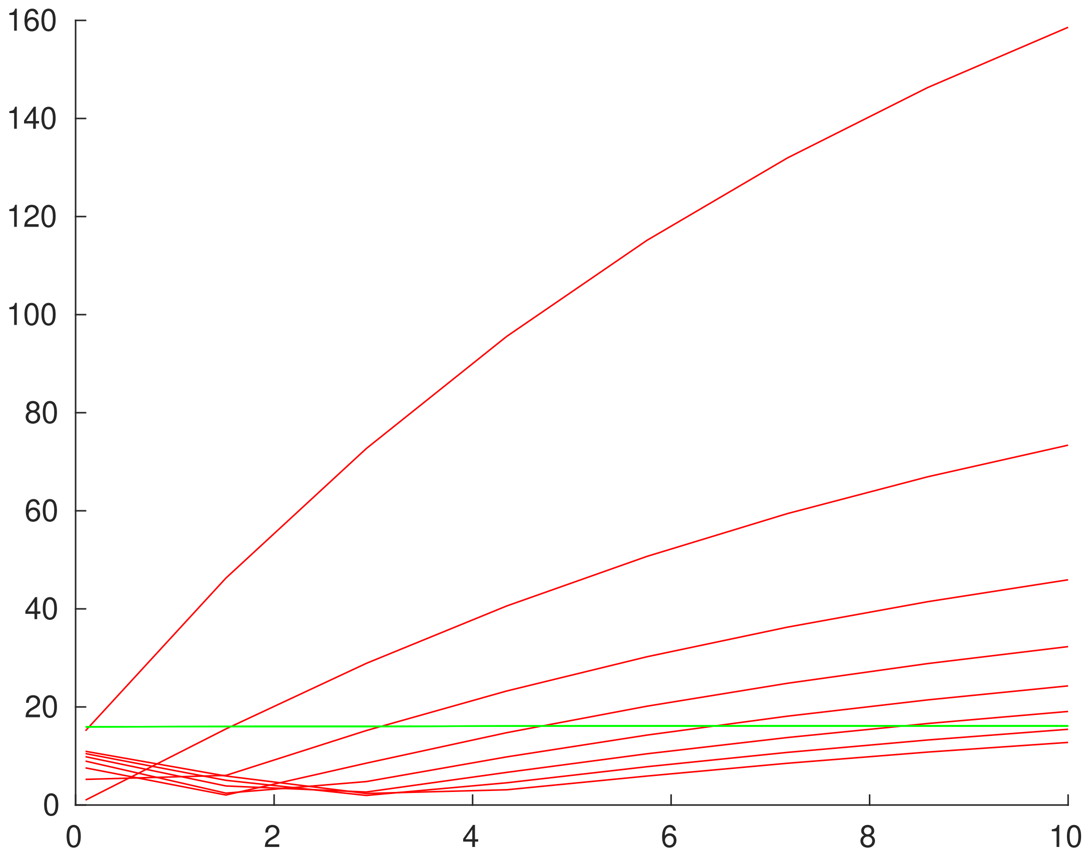

Figure 9: Graficele erorilor in functie de intrari

Subpunctul f
Cautam eroarea minima in vectorul calculat la subpunctul anterior, cat si indexul la care aceasta apare eroare:

```
[err_min, err_idx] = min(error_sys_2(i, :));
fprintf("Eroare min la j: %d => u2 = %d\n", err_idx, amp_var_u2(err_idx));
j_queue(i) = err_idx;
```

Subpunctul g
Calculam raspunsul partial "pe intervale":

```
out_1 = [];
out_2 = [];
last_i = 1;
last_x = [cih cih1 cih2 cih3 cih4];
for it=1:length(j_queue)
        j = j_queue(it);
        t = time_at(j);
        current_u1_data = [];
        current_u2_data = [];
        current_time = [];

        amp_u2 = amp_var_u2(j);
        u2_time = 0:time_step:time_count;
        u2_data = ones(1, length(u2_time)) * amp_u2;
        u2 = [u2_time' u2_data'];

        while (u1_time(last_i) < t)
                current_time(end + 1) = u1_time(last_i);
                current_u1_data(end + 1) = u1_data(last_i);
                current_u2_data(end + 1) = u2_data(last_i);
                last_i = last_i + 1;
        end
        [out, last_t, last_x] = lsim(linsys{j}, [current_u1_data; current_u2_data]', ...
                current_time', last_x(end, :));

        out_1 = [out_1; out(:, 3)];
        out_2 = [out_2; out(:, 5)];
end
```

Subpunctul h
Pentru a afla eroare de urmarire, calculam integrala:

```
fun = @(i) abs(nonlinear_out_2(int32(i)) - partial_linear_out_2(int32(i)));
err_2 = integral(fun, ti(1), ti(end));

fun = @(i) abs(nonlinear_out_4(int32(i)) - partial_linear_out_4(int32(i)));
err_4 = integral(fun, ti(1), ti(end))
```

```
fprintf("Erorare de urmarire y2 = %f\n", err_2);
fprintf("Erorare de urmarire y4 = %f\n", err_4);
```

Grafice f, g, h

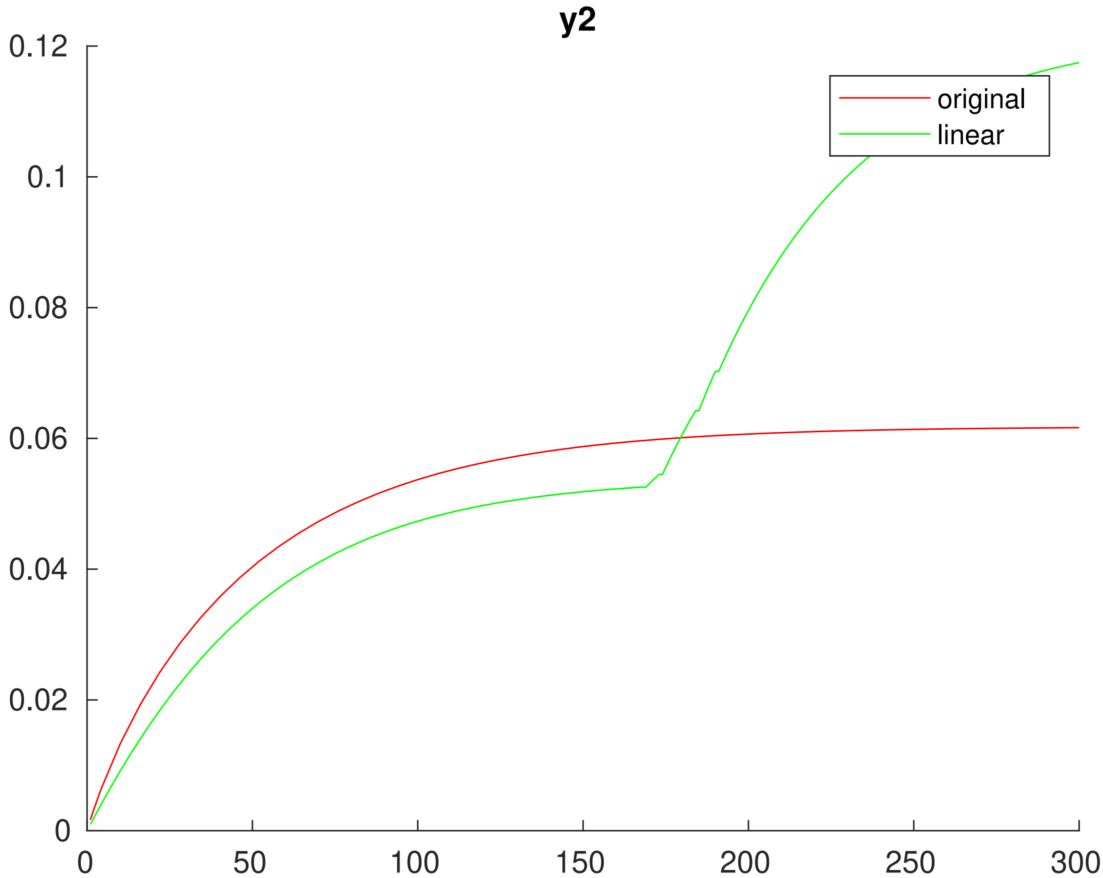

Figure 10: Evolutia iesirii h2 vs aproximarea sa liniara pe bucati

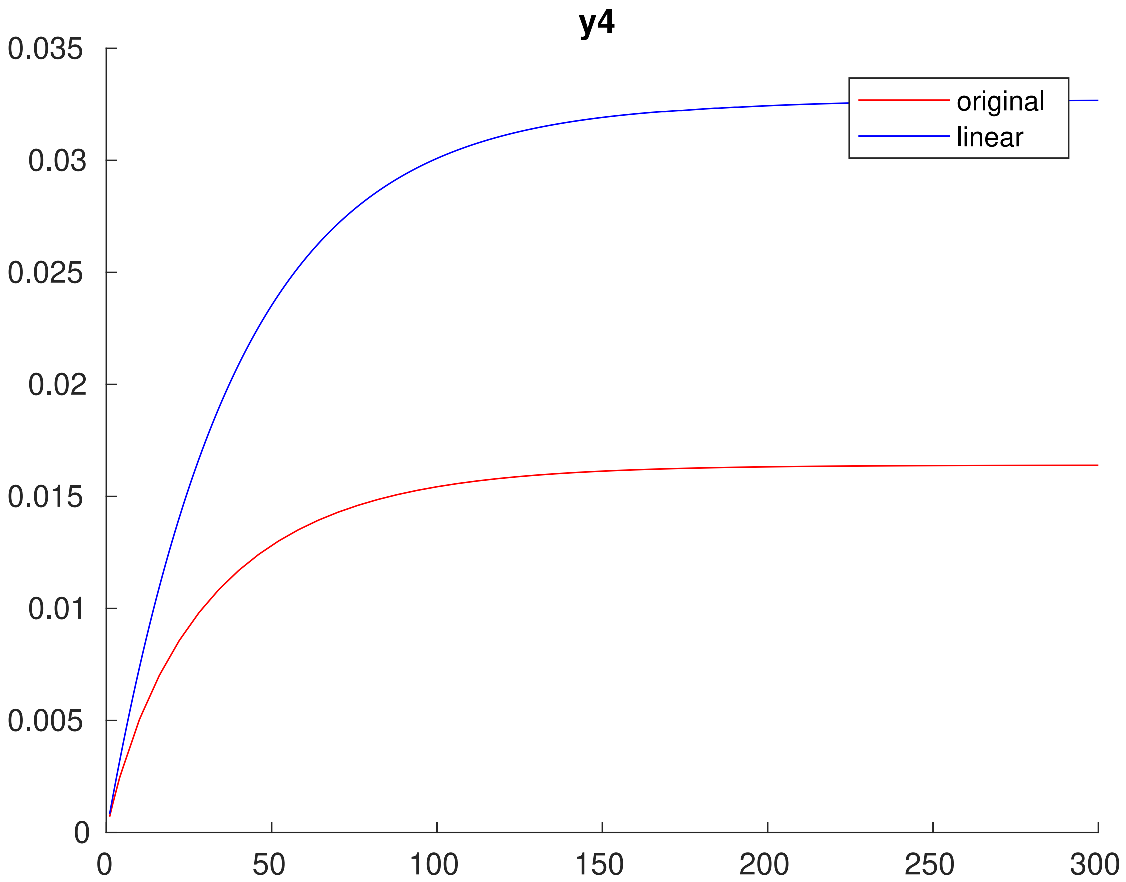

Figure 11: Evolutia iesirii h4 vs aproximarea sa liniara pe bucati

Model Simulink

Cod functie:

```
function [dh, dh1, dh2, dh3, dh4] = fcn(~, h1, h2, h3, h4,...
                u1, u2, a1, a2, a3, a4, ac, g, q, l1, l2, A1, A2, A3, A4, AT, kp)

    dh1 = 1/A1*(-a1*sqrt(2*g*h1) - ac*sign(q*g*h1 - q*g*h2)*sqrt(2*g*abs(h1 - h2)) + ...
            a3*sqrt(2*g*h3) + l1*kp*u1);
    dh2 = 1/A2*(-a2*sqrt(2*g*h2) - ac*sign(q*g*h1 - q*g*h2)*sqrt(2*g*abs(h1 - h2)) + ...
            a4*sqrt(2*g*h4) + l2*kp*u2);
    dh3 = 1/A3*(-a3*sqrt(2*g*h3) + (1 - l2)*kp*u2);
    dh4 = 1/A4*(-a4*sqrt(2*g*h4) + (1 - l1)*kp*u1);
    dh = 1/AT*(a1*sqrt(2*g*h1) + a2*sqrt(2*g*h2) - kp*u1 - kp*u2);
end
```

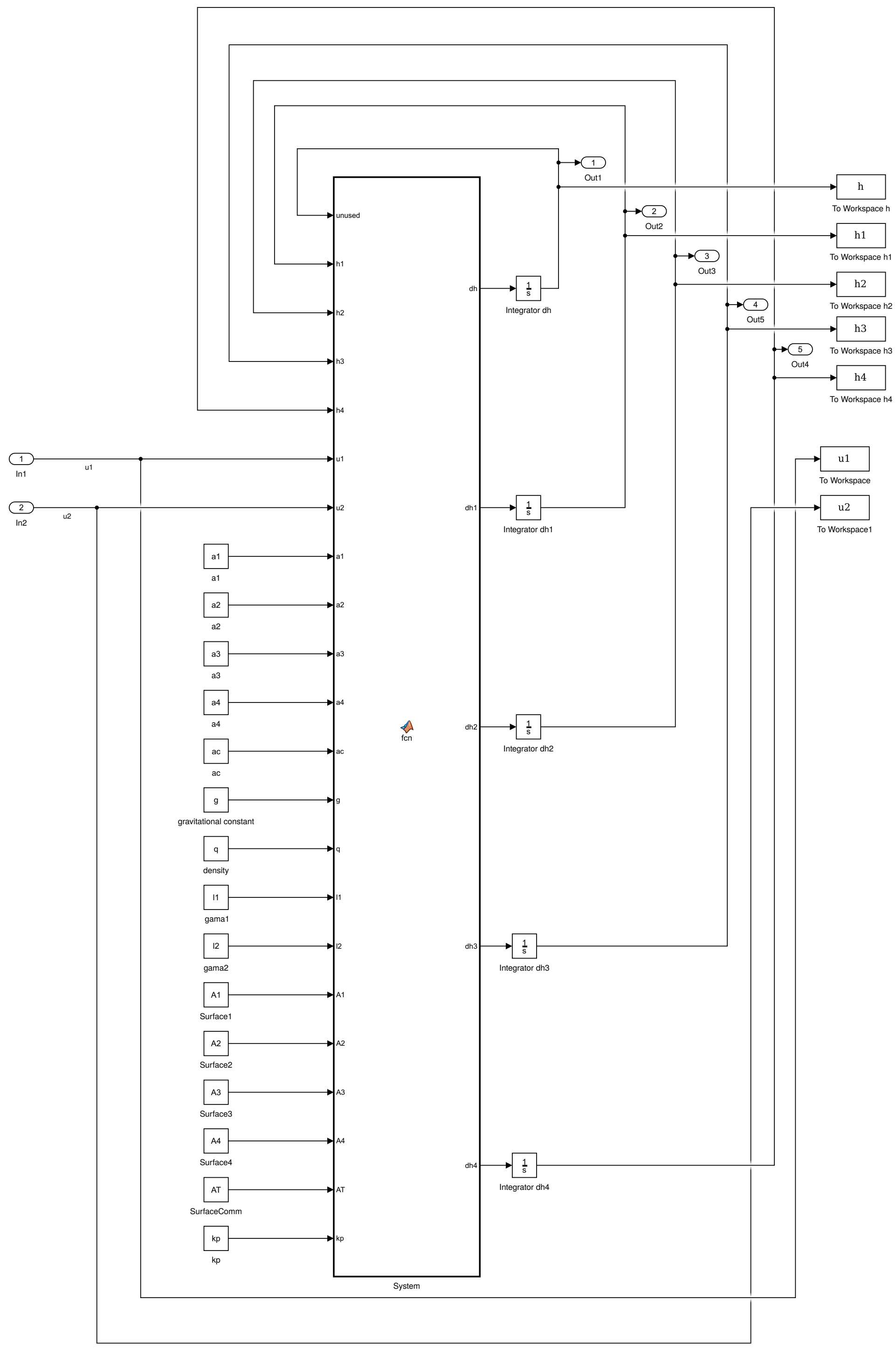

Figure 12: Diagrama bloc Simulink