Proiect Modelare și Simulare

Modelul mișcării unui satelit

Bolohan Marian-Cristian
333AB

December 2024

Cuprins

1 Descrierea modelului

Acest proiect constă în realizarea unui model format dintr-un satelit care orbitează Pământul și simularea acestuia pe baza cerințelor impuse. Pentru vectorul de poziție $r(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}^\top$ mișcarea acestui satelit poate fi aproximată cu soluţia sistemului de ecuaţii diferenţiale:

$$\ddot{r}(t) = \underbrace{-\frac{GM_\oplus}{\|r\|_2^2}\frac{r}{\|r\|_2} - \frac{3}{2}J_2 GM_\oplus \frac{R_\oplus^2}{\|r\|_2^5}\begin{bmatrix} \frac{x-5xz^2}{\|r\|_2^2} \\ \frac{y-5yz^2}{\|r\|_2^2} \\ \frac{3z-5z^3}{\|r\|_2^2} \end{bmatrix} + \omega_\oplus^2 \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + 2\omega_\oplus \begin{bmatrix} \dot{y} \\ -\dot{x} \\ 0 \end{bmatrix}}_{g(t,r(t),\dot{r}(t))}, r(t_0) = r_0, \dot{r}(t_0) = \dot{r}_0$$

unde constantele au valorile din Tabelul 1.1

Simbol	Semnificaţie	Valoare	Unitate
G	constanta gravitațională universală	$6.674 \cdot 10^{-11}$	$\mathrm{N\,m^2\,kg^{-2}}$
$M_\oplus$	masa Pământului	$5.972 \cdot 10^{24}$	kg
$R_\oplus$	raza Pământului	6371000	m
J_2	coeficient gravitaţional	$1.08262668 \cdot 10^{-3}$	–
$\omega_\oplus$	viteza unghiulară a Pământului	$7.2921 \cdot 10^{-5}$	$\mathrm{rad \cdot s^{-1}}$

Tabela 1.1: Valorile constantelor

1.1 Modelul dat

Se consideră doi sateliţi care orbitează Pământul, cu vectorii de poziţie $r_1(t)$ şi $r_2(t)$. Modelul este descris de:

$$\ddot{r}_1(t) = g(t, r_1(t), \dot{r}_1(t)) + u(t), \tag{1.1a}$$

$$\ddot{r}_2(t) = g(t, r_2(t), \dot{r}_2(t)), \tag{1.1b}$$

$$\dot{\varepsilon}(t) = \frac{\|r_1(t) - r_2(t)\|_2}{\|r_2(t)\|_2}, \tag{1.1c}$$

$$y(t) = \varepsilon(t). \tag{1.1d}$$

Condițiile inițiale sunt:

$$r_1(t_0) = 10^6 \cdot \begin{bmatrix} -3.111566746661099 & 2.420733547442338 & -5.626803092559423 \end{bmatrix}^\top,$$

$$\dot{r}_1(t_0) = 10^3 \cdot \begin{bmatrix} 4.953572247000772 & -3.787243278806948 & -4.362500902062312 \end{bmatrix}^\top,$$

$$r_2(t_0) = 10^6 \cdot \begin{bmatrix} -3.422723421327209 & 2.662806902186572 & -6.189483401815366 \end{bmatrix}^\top,$$

$$\dot{r}_2(t_0) = 10^3 \cdot \begin{bmatrix} 5.448929471700850 & -4.165967606687643 & -4.798750992268544 \end{bmatrix}^\top.$$

2 Etapele de realizare a proiectului

2.1 Cerința 1

În cadrul primei cerințe am ales condiția inițială:

$$\varepsilon(t_0) = 10^4$$

De asemenea, am ales și valorea $k = 5$ definită în semnalul exogen $u(t) = k \cdot 10^{-3} \cdot \mathbf{1}(t)$.

```
clc; clear; close all
load valori.mat
%% Cerinta 1
epsilon0 = 1e4;
k = 5;
```

2.2 Cerința 2

Implementrea modelului folosind blocuri `Integrator` și `Matlab Function` este prezentată în Figura 2.1.

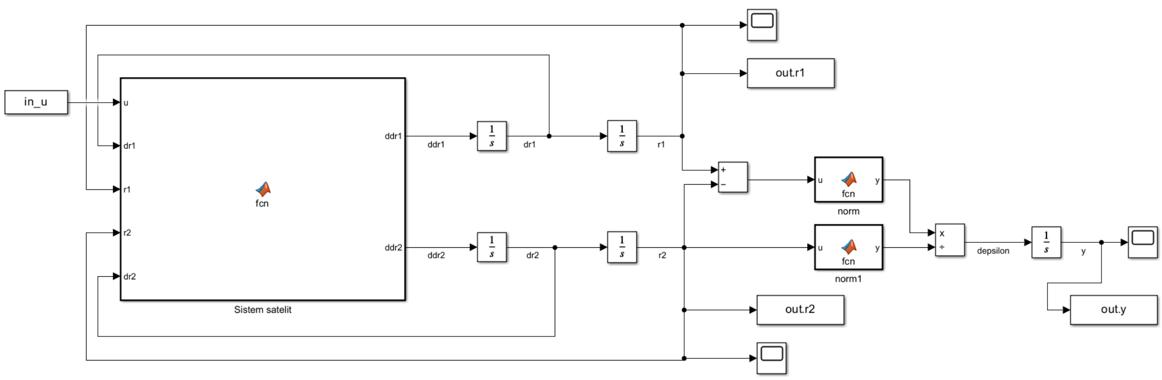

Figura 2.1: Implementarea modelului folosind blocuri `Integrator` și `Matlab Function`

Implentarea blocului `Matlab Function` numit *Sistem satelit* a fost realizată folosind următorul cod:

```
function [ddr1, ddr2]= fcn(u, dr1, r1, r2, dr2)

G = 6.674e-11;
Mp = 5.972e24;
Rp = 6371000;
J2 = 1.08262668e-3;
omegap = 7.2921e-5;

ddr1 = -((G*Mp)/norm(r1)^2) .* (r1/norm(r1)) - ...
       (3/2 * J2 * G * Mp * Rp^2 / norm(r1)^5) .* ...
```

```
        [(r1(1) - 5 * r1(1) * r1(3)^2)/norm(r1)^2;
         (r1(2) - 5 * r1(2) * r1(3)^2)/norm(r1)^2;
         (r1(3) - 5 * r1(3) * r1(3)^2)/norm(r1)^2] + ...
         omegap^2 .* [r1(1); r1(2); 0] + ...
         2*omegap .* [dr1(2); -dr1(1); 0] + ...
         u;

ddr2 = -((G*Mp)/norm(r2)^2) .* (r2/norm(r2)) - ...
        (3/2 * J2 * G * Mp * Rp^2 / norm(r2)^5) .* ...
        [(r2(1) - 5 * r2(1) * r2(3)^2)/norm(r2)^2;
         (r2(2) - 5 * r2(2) * r2(3)^2)/norm(r2)^2;
         (r2(3) - 5 * r2(3) * r2(3)^2)/norm(r2)^2] + ...
         omegap^2 .* [r2(1); r2(2); 0] + ...
         2*omegap .* [dr2(2); -dr2(1); 0];
```

iar a blocurilor numite *Norm* și *Norm1*:

```
function y = fcn(u)
y = norm(u);
```

Ieșirea sistemului la intrarea $u(t)$, pentru timpul de simulare $Tmax = 50000$, este ilustrată în Figura 2.2:

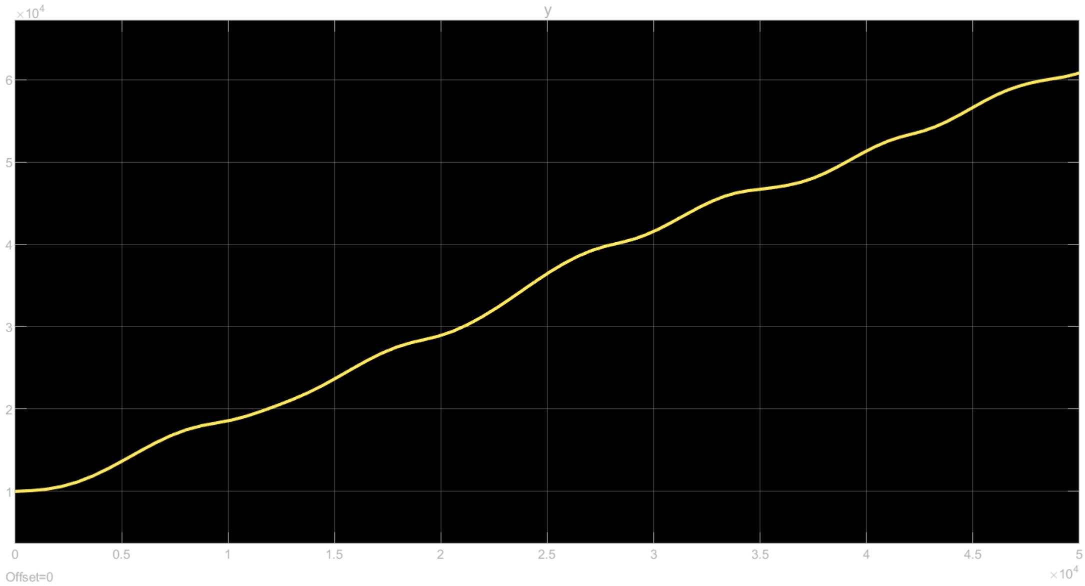

Figura 2.2: Ieșirea sistemului la intrarea $u(t)$

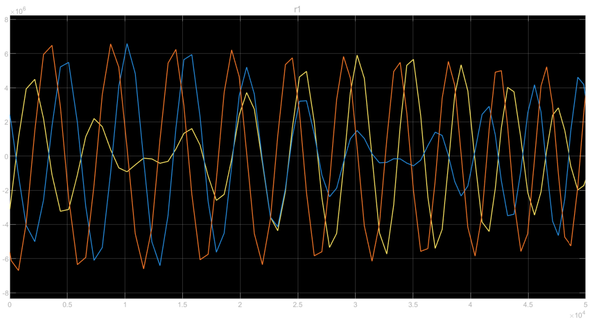

Figura 2.3: Vizualizare r_1

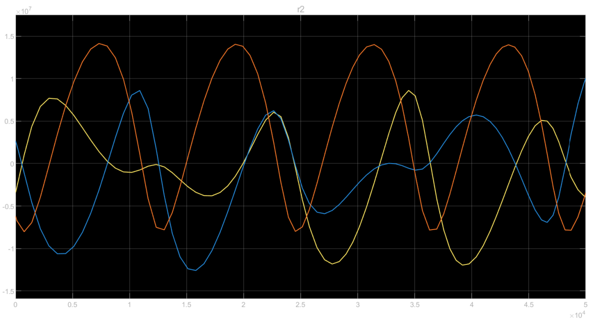

Figura 2.4: Vizualizare r_2

2.3 Cerința 3

Pentru a rearanja sistemul de ecuații sub forma unui sistem de ecuaţii diferenţiale de ordinul I, i.e. $\frac{d}{dt}x(t) = f(t, x(t))$, am realizat următoarele calcule:

$$\ddot{r}(t) = g(t, r(t), \dot{r}(t))$$

$$x(t) = \begin{bmatrix} r(t) \\ \dot{r}(t) \end{bmatrix} \Rightarrow \dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ g(t, r(t), \dot{r}(t)) \end{bmatrix} \tag{2.1}$$

Ecuația 2.1 este modelul pentru un singur satlit. Atunci modelul pentru cei doi sateliți este dat de ecuația 2.2.

$$\dot{x}(t) = \begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \\ \dot{\varepsilon}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \dot{r_1}(t) \\ g(t, r_1(t), \dot{r_1}(t)) \\ \dot{r_2}(t) \\ g(t, r_2(t), \dot{r_2}(t)) \\ \frac{\|r_1(t) - r_2(t)\|_2}{\|r_2(t)\|_2} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot u(t)}_{f(t, x(t))} \tag{2.2}$$

Codul Matlab necesar implementarii acestei funcții este:

```
%% Cerinta 3
g = @(t, r, dr) -((G*Mp)/norm(r)^2) .* (r/norm(r)) - ...
       (3/2 * J2 * G * Mp * Rp^2 / norm(r)^5) .* ...
       [(r(1) - 5 * r(1) * r(3)^2)/norm(r)^2;
        (r(2) - 5 * r(2) * r(3)^2)/norm(r)^2;
        (r(3) - 5 * r(3) * r(3)^2)/norm(r)^2] + ...
        omegap^2 .* [r(1); r(2); 0] + ...
        2*omegap .* [dr(2); -dr(1); 0];

f = @(t, x) [x(4:6); g(t, x(1:3), x(4:6)); x(10:12); g(t, x(7:9), x(10:12));
    norm(x(1:3) - x(7:9))/norm(x(7:9))];
```

2.4 Cerința 4

Pentru a rezolva sistemul diferenția rezultat anterior, am implementat metoda **Runge-Kutta** de ordinul patru:

$$x(t+h) \approx x(t) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

$$k_1 = f(t, x(t)),$$

$$k_2 = f\left(t + \frac{h}{2}, x(t) + \frac{hk_1}{2}\right),$$

$$k_3 = f\left(t + \frac{h}{2}, x(t) + \frac{hk_2}{2}\right),$$

$$k_4 = f\left(t + h, x(t) + hk_3\right).$$

având implementarea în Matlab:

```
%% Cerinta 4
% pas de integrare si Tmax
Tmax = 50000;
h = 1;
% variabile de start
x = [r1_0; dr1_0; r2_0; dr2_0; epsilon0];
```

```
t = 0;
num_steps = ceil(Tmax / h);
X = zeros(size(x, 1), num_steps);
T = zeros(1, num_steps);
% Metoda Runge-Kutta
i = 1;
while t < Tmax
    k1 = f(t, x);
    k2 = f(t + h/2, x + h*k1/2);
    k3 = f(t + h/2, x + h*k2/2);
    k4 = f(t + h, x + h*k3);

    X(:, i) = x;
    T(i) = t;
    x = x + h/6 * (k1 + 2*k2 + 2*k3 + k4);
    t = t + h;
    i = i + 1;
end

r1 = X(1:3, :);
r2 = X(7:9, :);
```

Am comparat ulterior, rezultatele metodei cu cele din Simulink, din care reiese faptul că metoda este eficientă, nu există diferențe vizibile între cele două rezultate:

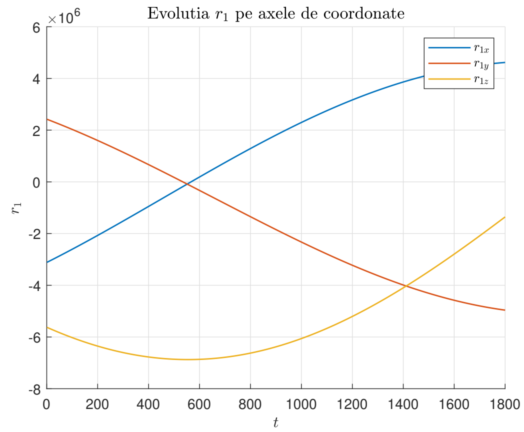

Figura 2.5: Rezultate Runge-Kutta pentru r_1

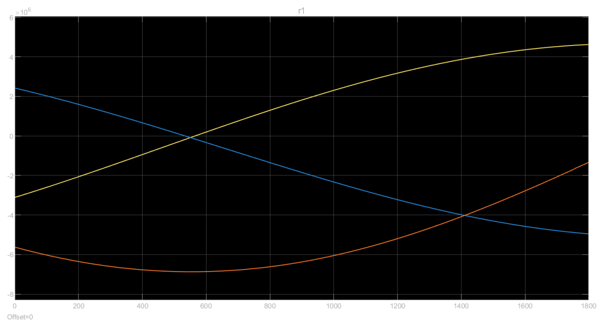

Figura 2.6: Rezultate Simulink pentru r_1

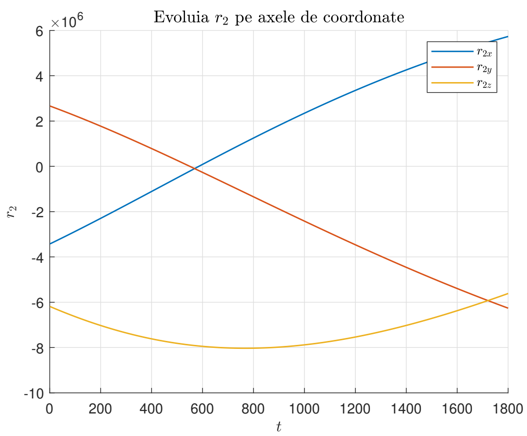

Figura 2.7: Rezultate Runge-Kutta pentru r_2

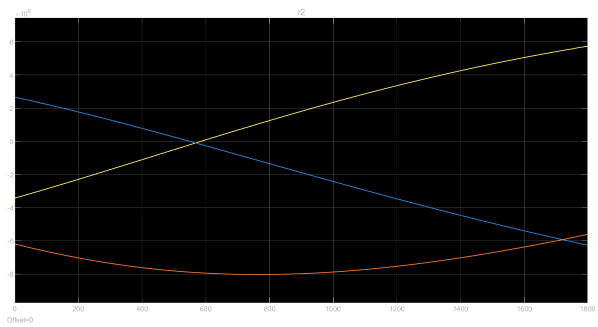

Figura 2.8: Rezultate Simulink pentru r_1

2.5 Cerința 5

Pentru a putea vizualiza în Matlab orbita sateliților r_1 și r_2, am folosit următoarea secvența de cod, unde a fost definit și semnalul de intrare $u(t)$:

```
% Simulink
mdl = 'satelit_mdl';
% intrarea
t = 0:h:Tmax;
u = k * 1e-3 * double(t>=0);
in_u = timeseries(u, t);

% iesirea
load_system(mdl);
set_param(mdl, "StopTime", num2str(Tmax));
out = sim(mdl);

slx_r1 = squeeze(out.r1.Data);
slx_r2 = squeeze(out.r2.Data);
```

Nu au existat diferențe vizibile între cele două metode, pentru $Tmax = 1800$:

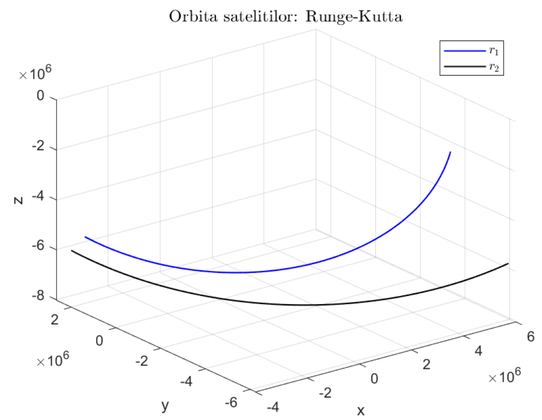

Figura 2.9: Rezultate Runge-Kutta

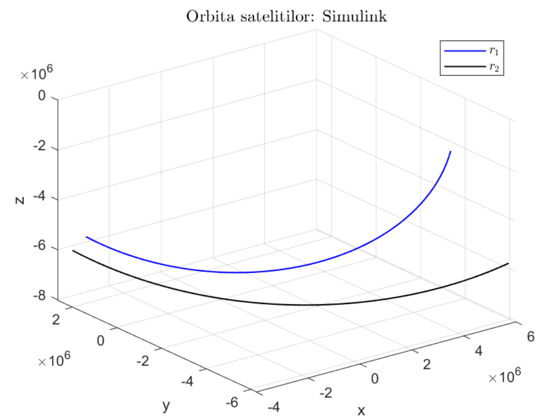

Figura 2.10: Rezultate Simulink

Pentru $Tmax = 50000$, apar anumite diferențe datorate timpului de eșantionare dim Simulink:

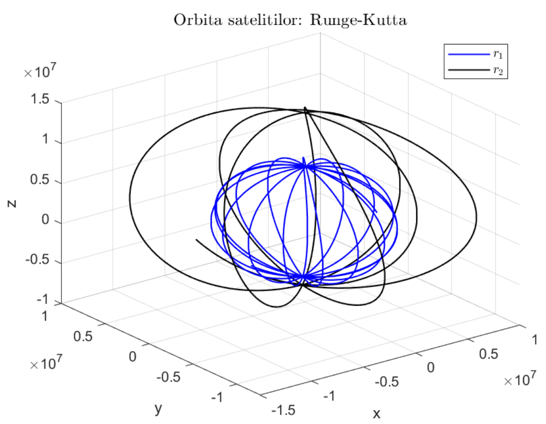

Figura 2.11: Rezultate Runge-Kutta

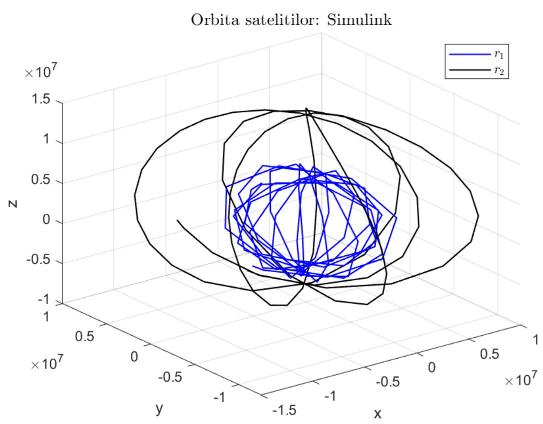

Figura 2.12: Rezultate Simulink

Pentru a rezolva aceste diferențe care apar la timpi mari de simulare, am interpolat ieșirea din Simulink:

```
% Interpolare rezultat Simulink, pentru un plot mai fin
% Acest lucru este vizibil la Tmax mare
num_points = size(slx_r1, 2);
t_original = linspace(0, 1, num_points);
t_fine = linspace(0, 1, num_points * 10);

slx_r1_smooth = interp1(t_original, slx_r1', t_fine, 'spline')';
slx_r2_smooth = interp1(t_original, slx_r2', t_fine, 'spline')';
```

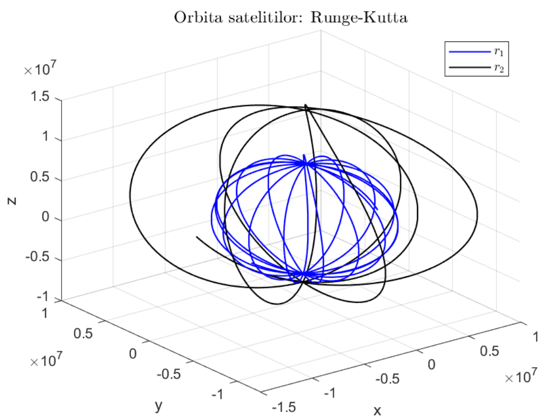

Figura 2.13: Rezultate Runge-Kutta

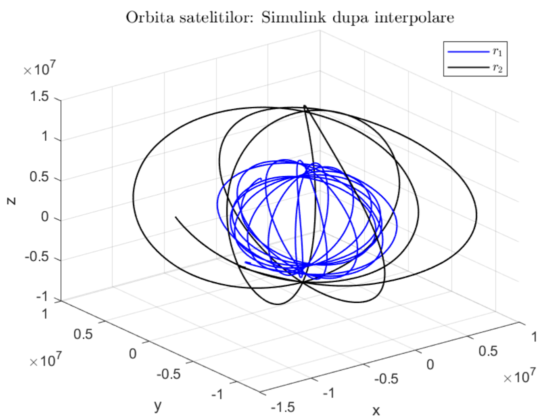

Figura 2.14: Rezultate Simulink după interpolare

2.6 Cerința 6

Pentru a plota ieșirea $y(t) = \varepsilon(t)$, am extras ultimul element din vectorul de stare, pentru metoda Runge-Kutta.
Pentru ieșirea din Simulink, am interpolat rezultatul pe întreg suprotul de timp pentru a obține un grafic relevant:

```
%% Cerinta 6
y_rk = X(13, :);

y_slx = squeeze(out.y.Data);

% interpolare
num_points = size(y_rk, 2);
t_original = linspace(0, 1, size(y_slx, 1));
t_fine = linspace(0, 1, num_points);
y_slx_smooth =  interp1(t_original, y_slx', t_fine, 'spline');
```

Eroarea de intregrare a fost calculată ca:

$$\epsilon(t) = ||y^{Slx}(t) - y^{RK}(t)||_2$$

```
e = vecnorm(y_rk - y_slx_smooth, 2, 1);
```

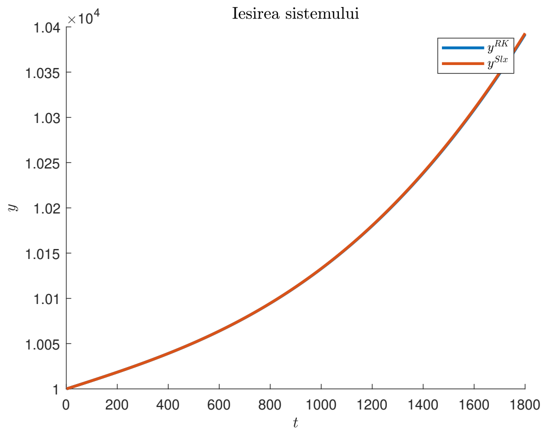

Figura 2.15: Ieșirile rezultate prin cele două metode

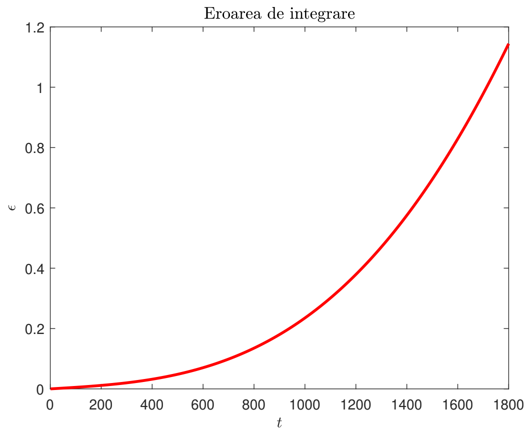

Figura 2.16: Eroarea de intregrare

Se poate observa că eroarea de integrare este foarte mică, la momentul final $Tmax = 1800$, acesta având valoarea, $\epsilon = 1.2$, aproximativ 0.01% din valoarea finală a lui y^{RK}.

2.7 Cerința 7

Pentru a trasa caracteristica statică $\varepsilon^*(u^*)$, am variat parametrul k, de care depinde u, pentru a crea mai multe semnale u^*.

$$k = \begin{bmatrix} 5 & 10 & 50 & 100 & 250 & 500 \end{bmatrix}$$

```
t = 0:h:Tmax;
k_star = [5, 10, 50, 100, 250, 500]; % pentru u_star
epsilon_star = zeros(1, numel(k_star));
u_star = zeros(1, numel(k_star));

for i = 1:numel(k_star)
    u = k_star(i) * 1e-3 * double(t>=0);
    in_u = timeseries(u, t);

    out = sim(mdl);
    epsilon_out = squeeze(out.y.Data);

    epsilon_star(i) = epsilon_out(end);
    u_star(i) = u(end);
end
```

Caracteristica statică rezultată este ilustrată în Figura 2.17

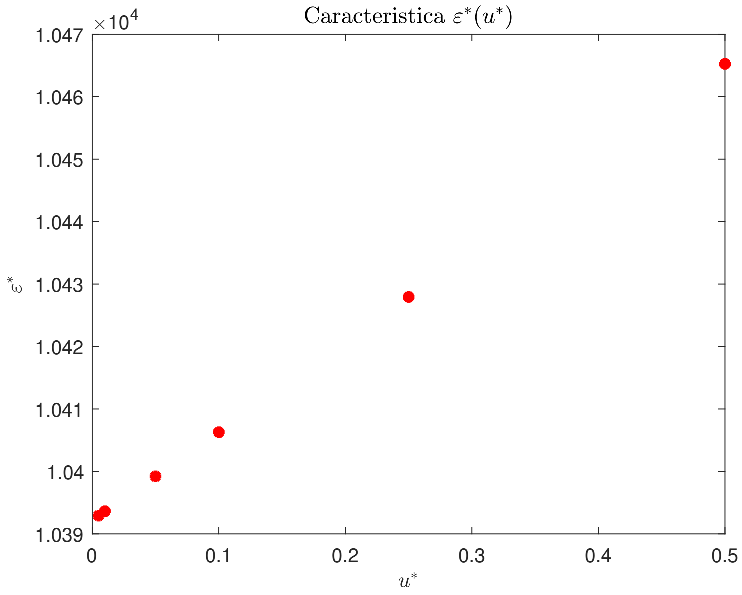

Figura 2.17: Caracteristica statică $\varepsilon^*(u^*)$

2.8 Cerința 8

Din Figura 2.17 se observă liniaritatea caracteristicii, astfel pentru determinarea polinomului de aproximare, am folosit funcția `polyfit`, pentru a găsi o funcție de gradul I care aproximează cel mai bine datele în sensul celor mai mici pătrate.

```
p = polyfit(u_star, epsilon_star, 1);

epsilon_aprox = polyval(p, u_star);
```

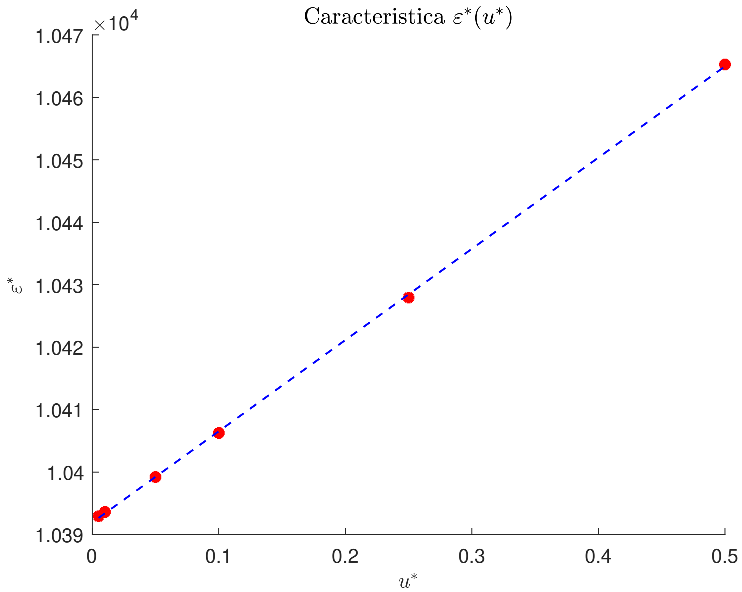

Figura 2.18: Caracteristica statică $\varepsilon^*(u^*)$

2.9 Cerința 9

Folosind secvența următoare de cod, am realizat 100 de simulări pentru a urmări evoluția lui $y(t)$, unde condiția inițială $r_2(t_0)$ depinde de relația:

$$\tilde{r_2}(t_0) = (1 + \alpha) \cdot r_2(t_0) \text{ , unde } \alpha \sim \mathcal{N}(0, 0.1)$$

```
%% Cerinta 9
load valori.mat % pentru a reseta valoarea lui r2_0 inainte de simulari
% folosim intrarea aleasa la cerinta 1
t = 0:h:Tmax;
u = k * 1e-3 * double(t>=0);
in_u = timeseries(u, t);

% N(0, 0.1)
mu = 0; % media
var = 0.1; % varianta

r2_0_copy = r2_0;
nr_iter = 100;
sample_size = Tmax;
y = zeros(nr_iter, sample_size);

figure;
hold on;
for i = 1:nr_iter
    alpha = sqrt(var) .* randn(1, 1) + mu;
    r2_0 = (1 + alpha) .* r2_0_copy;

    out = sim(mdl);

    y_slx = squeeze(out.y.Data)';

    % interpolare pentru a corespunde cu suportul de timp
    t_original = linspace(0, 1, size(y_slx, 2));
    t_fine = linspace(0, 1, sample_size);
    y_slx_smooth =  interp1(t_original, y_slx', t_fine, 'spline');

    y(i, :) = y_slx_smooth;

    plot(y(i, :), ':', 'HandleVisibility','off');
end

y_mean = mean(y, 1);
```

Rezultatul este ilustrat în Figura 2.19

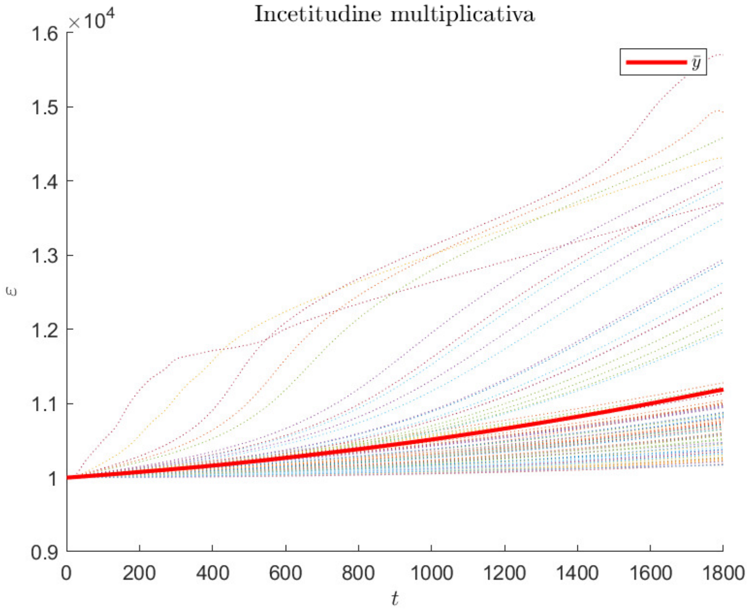

Figura 2.19: Ieșirile $y(t)$

În continuare, am ales un prag dat de ultima valoare a mediei ieșirii $\bar{y}$ pentru a calcula probabilitatea ca distanța relativă totală dintre sateliți $\varepsilon(t)$ să depășească acest prag. Pentru aceste simulari pragul are valoarea de $1.1187 \cdot 10^4$. Astfel:

```
Probabilitatea ca ε(t) să depășească y_{mean}(end) = 11187.09 este 22.00%
```

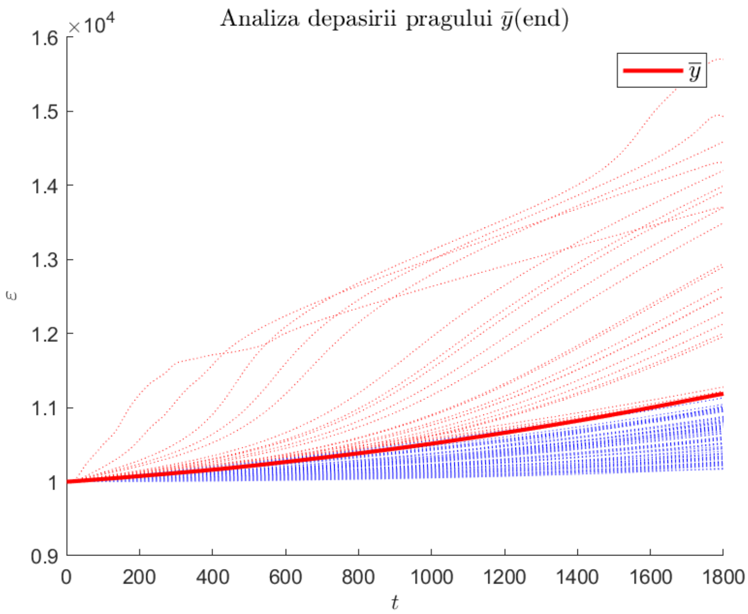

Figura 2.20: Ieșirile $y(t)$ care depășesc pragul

2.10 Cerința 10

Folosind secvența următoare de cod, am realizat 100 de simulări pentru a urmări evoluția lui $y(t)$, unde condiția inițială $r_2(t_0)$ depinde de relația:

$$\tilde{r_2}(t_0) = \alpha + r_2(t_0) \text{ , unde } \alpha \in \mathbb{R}^3 \text{ și } \alpha_i \sim \mathcal{N}(0,5), i = 1:3$$

```
%% Cerinta 9
load valori.mat % pentru a reseta valoarea lui r2_0 inainte de simulari
% folosim intrarea aleasa la cerinta 1
t = 0:h:Tmax;
u = k * 1e-3 * double(t>=0);
in_u = timeseries(u, t);

% N(0, 5)
mu = 0; % media
var = 5; % varianta

r2_0_copy = r2_0;

nr_iter = 100;
sample_size = Tmax;
y = zeros(nr_iter, sample_size);

figure;
hold on;
for i = 1:nr_iter
    alpha = sqrt(var) .* randn(numel(r2_0), 1) + mu;
    r2_0 = alpha + r2_0_copy;

    out = sim(mdl);

    y_slx = squeeze(out.y.Data)';

    % interpolate pentru outputuri mai mici decat sample_size
    num_points = size(y_slx, 2);
    t_original = linspace(0, 1, size(y_slx, 2));
    t_fine = linspace(0, 1, sample_size);
    y_slx_smooth =  interp1(t_original, y_slx', t_fine, 'spline');

    y(i, :) = y_slx_smooth;

    plot(y(i, :), ':', 'HandleVisibility','off');
end

y_mean = mean(y, 1);
```

Incetitudinea aditivă contribuie foarte puțin, aproape negiljabil, la valorile ieșirii $y(t)$. Astfel, pentru o mai bună vizualizare a graficelor, am folosit comanda `xlim([1008.782 1008.795])`.

Rezultatul este ilustrat în Figura 2.21

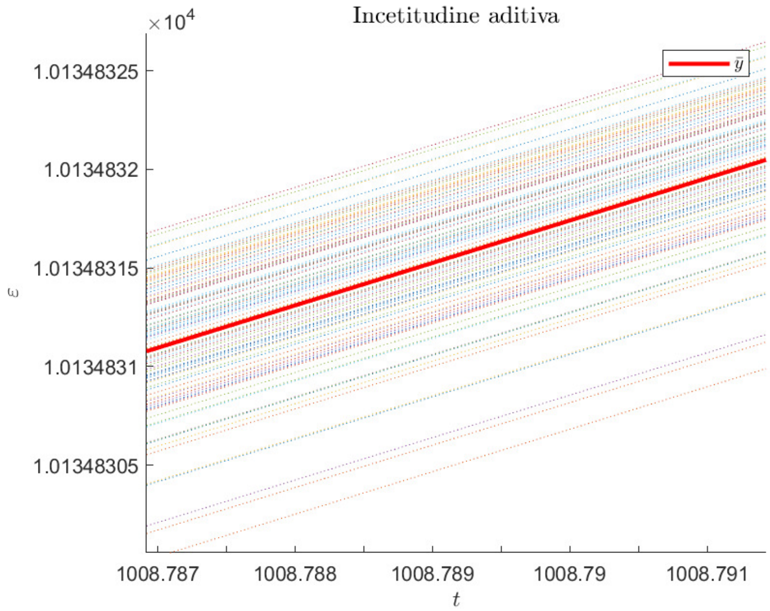

Figura 2.21: Ieșirile $y(t)$

În continuare, am ales un prag dat de ultima valoare a mediei ieșirii $\bar{y}$ pentru a calcula probabilitatea ca distanța relativă totală dintre sateliți $\varepsilon(t)$ să depășească acest prag. Pentru aceste simulari pragul are valoarea de $1.0393 \cdot 10^4$. Astfel:

```
Probabilitatea ca ε(t) să depășească y_{mean}(end) = 10392.93 este 52.00%
```

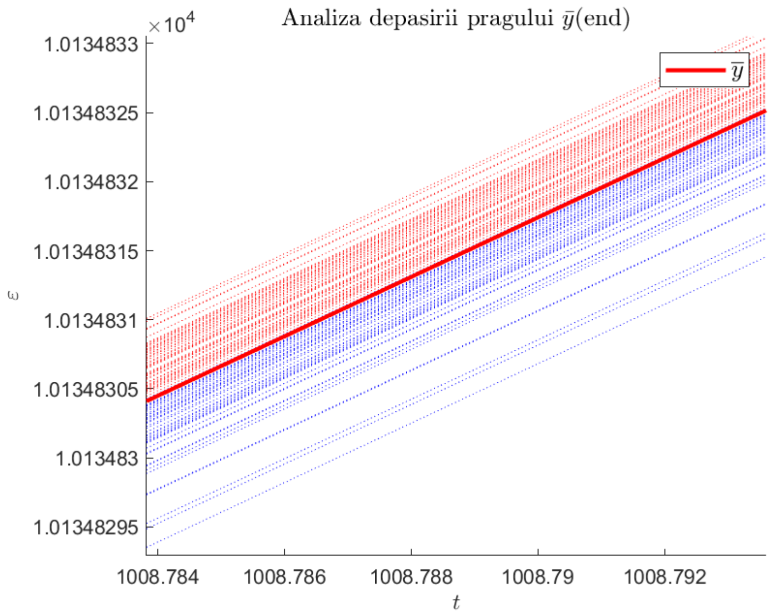

Figura 2.22: Ieșirile $y(t)$ care depășesc pragul

2.11 Cerința 11

Pentru început am creat semnalul exogen $u(t)$, care are o distribuție normală: $u(t) \sim \mathcal{N}(0, 1)$. Ulterior am apelat modelul din Simulink cu semnalul obținut.

```
%% Cerinta 11
t = 0:1:Tmax; t = t';
sample_size = Tmax;

u = randn(numel(t), 1); % apartine N(0,1)
in_u = timeseries(u, t);

out = sim(mdl);

y_slx = squeeze(out.y.Data)';

% interpolare pentru outputuri mai mici decat sample_size
num_points = size(y_slx, 2);
t_original = linspace(0, 1, size(y_slx, 2));
t_fine = linspace(0, 1, sample_size);
y_slx_smooth =  interp1(t_original, y_slx', t_fine, 'spline');

second_dy = diff(y_slx_smooth, 2);
```

Se observă în Figura 2.23, că a doua derivată discretă a ieșirii are valori foarte mici, motiv pentru care a fost utilizată comanda `ylim([-0.005 0.005])`.

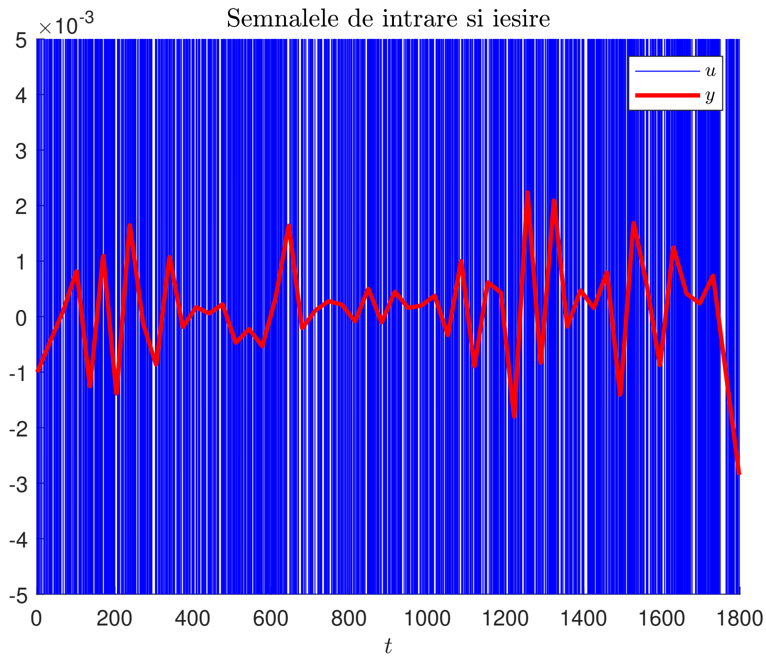

Figura 2.23: Semnalul de intrare și a doua derivată discretă a ieșirii

Se observă că semnalul rezultat are aproximativ aceiași medie cu semnalul exogen, iar deviația standard este cu mult mai mică decat cea a intrarii:

$$media = 1.3140 \cdot 10^{-4} \approx 0$$

$$std = 6.6946 \cdot 10^{-4} \ll 1$$