Adrian DĂNILĂ

MODELAREA ŞI IDENTIFICAREA SISTEMELOR DINAMICE

Culegere de probleme rezolvate în mediul software

Python Sympy

Departamentul de Automatică şi Tehnolgia informaţiei
Universitatea *Transilvania* din Braşov

MODELAREA ŞI IDENTIFICAREA SISTEMELOR DINAMICE

Culegere de probleme rezolvate în mediul software Python Sympy

Adrian DĂNILĂ

EDITURA UNIVERSITĂȚII TRANSILVANIA DIN BRAȘOV

2022

Prefață

Rezolvarea de exerciții și probleme este o etapă necesară pentru aprofundarea cunoștințelor de specialitate, mai ales în cadrul activităților disciplinelor din domeniile tehnice.

Această lucrare conține o selecție de probleme de seminar, probleme pregătitoare de laborator și probleme de examen care au fost propuse la disciplinele Identificarea sistemelor - anul III specializarea Automatică și Informatică aplicată și Modelare și simulare - anul III specializarea Tehnologia informației, în perioada 2012 - 2022.

Structura pe capitole a lucrării respectă structura cursului Modelarea și Identificarea sistemelor dinamice, Editura Universității Transilvania din Brașov, 2013 - pe care o recomand spre consultare ca suport teoretic pentru rezolvarea problemelor.

Fiecare capitol al lucrării este organizat astfel încât să faciliteze lucrul independent al rezolvitorului: în prima parte sunt prezentate enunțurile problemelor, în partea a doua sunt răspunsurile iar în partea finală sunt prezentate, în detaliu, rezolvările.

Toate problemele sunt rezolvate în mediul de programare Python, modulul de calcul simbolic Sympy ceea ce permite cititorului să se concentreze mai mult asupra organizării pe etape a rezolvării problemei respectiv asupra interpretării rezultatelor calculelor și mai puțin asupra aspectelor de calcul analitic și numeric.

Recomandare: *Aplicațiile sunt scrise și rulate în mediul de calcul web Jupyter Notebook. Fișierele care conțin datele experimentale trebuie încărcate în aceeași locație în care se rulează aplicația.*

Brașov, septembrie 2022.

Cuprins

Prefaţă **3**

1 Introducere în Identificarea sistemelor **7**

2 Variabile aleatoare şi funcţii de probabilitate **35**

3 Procese aleatoare şi zgomote **45**

4 Procese dinamice liniare **67**

5 Metode neparametrice **87**

6 Metode parametrice **125**

Bibliografie **157**

1

Introducere în Identificarea sistemelor

Enunțuri

Problema 1.1

Fiind dat un proces dinamic proporțional cu întârziere de ordinul unu, $PT1$ reprezentat prin funcția de transfer:

$$G(s) = \frac{K_a}{T_f \cdot s + 1},$$

în care K_a este factorul de proporționalitate și T_f este constanta de timp. Se cer următoarele:

(a) să se calculeze expresia funcției pondere $g(t)$ a procesului dinamic;

(b) să se calculeze expresia funcției indiciale $g_1(t)$ a procesului dinamic;

(c) ce relație este între cele două funcții?

Problema 1.2

Pe baza rezultatelor de la problema precedentă, să se demonstreze următoarele proprietăți ale funcțiilor pondere și indicială ale proceselor de tip $PT1$:

(a) valoarea în origine a funcției pondere $g(0_+)$ este numeric egală cu raportul $\frac{K_a}{T_f}$;

(b) fiind date două valori distincte $g(t_1)$ și $g(t_2)$ ale funcției pondere între care există relația $\frac{g(t_1)}{g(t_2)} = e$ (numărul lui Euler) atunci diferența dintre argumentele omoloage ale funcției este numeric egală cu constanta de timp, $t_2 - t_1 = T_f$;

(c) tangenta la graficul funcției pondere $g(t)$ care trece prin punctul de coordonate $(0; g(0_+))$ intersectează axa absciselor în punctul de coordonate $(T_f; 0)$;

(d) aria suprafeței cuprinsă între graficul funcției pondere și axa orizontală este numeric egală cu K_a.

(e) valoarea la infinit a funcției indiciale este numeric egală cu factorul de proporționalitate - numit și câștig static - al funcției de transfer.

Problema 1.3

În tabelele (1.1) și (1.2) se dau eșantioanele unei secvență de 20 de valori ale funcției pondere observată la portul de ieșire al unui sistem dinamic. Eșantioanele secvenței nu sunt perturbate de zgomot aditiv pe linia de măsurare.
Se formulează ipoteza că procesul dinamic asociat este de tip $PT1$.
Folosind rezultatele de la problema 1.2, se cere să se estimeze valorile parametrilor funcției de transfer ai procesului dinamic.

Tabela 1.1: Valorile variabilei timp în exempul din problema 1.3.

0.000 0.056 0.111 0.167 0.222 0.278 0.333 0.389 0.444 0.500 0.556 0.611 0.667 0.722 0.778 0.833 0.889 0.944 1.000

Tabela 1.2: Valorile funcției pondere în exempul din problema 1.3.

25.000 18.937 14.344 10.865 8.230 6.234 4.722 3.577 2.709 2.052 1.554 1.177 0.892 0.676 0.512 0.388 0.294 0.222 0.168

Problema 1.4

Un proces dinamic proporțional cu întârziere de ordinul doi supra-amortizat, $PT2$ este reprezentat prin funcția de transfer:

$$G(s) = \frac{K_a}{T_{f1}T_{f2} \cdot s^2 + (T_{f1} + T_{f2}) \cdot s + 1},$$

în care K_a este constanta de proporționalitate și T_{f1}, T_{f2} sunt constantele de timp ale procesului.
Se cer următoarele:

(a) să se calculeze expresia funcției pondere $g(t)$ a procesului dinamic;

(b) să se calculeze expresia funcției indiciale $g_1(t)$ a procesului dinamic;

(c) ce relație există între expresiile funcțiilor pondere și indicială?

Problema 1.5

Să se demonstreze următoarele proprietăți ale funcțiilor pondere și indicială ale proceselor dinamice de tip $PT2$ supra-amortizate:

(a) aria suprafeței cuprinsă între graficul funcției pondere și axa orizontală este numeric egală cu factorul de proporționalitate al funcției de transfer;

(b) abscisa punctului de maxim pe graficul funcției pondere nu depinde de factorul de proporționalitate al funcției de transfer;

(c) valoarea la infinit a funcției indiciale este numeric egală cu factorul de proporționalitate al funcției de transfer;

(d) fiind dată o deplasare în domeniul timpului, τ, funcția pondere verifică relația:

$$g(t+2\tau) - (a_1 + a_2) \cdot g(t+\tau) + a_1 \cdot a_2 \cdot g(t) = 0, \ \ \forall t \in \mathbb{R}_+;$$

în care $a_1 = e^{\frac{-\tau}{T_1}}$ și $a_2 = e^{\frac{-\tau}{T_2}}$.

Problema 1.6

În tabelele (1.3) și (1.4) se dau eșantioanele unei secvență de 100 de valori ale funcției pondere observată la portul de ieșire al unui sistem dinamic. Perioada de eșantionare este $T_e = 0.01$ secunde. Eșantioanele secvenței nu sunt perturbate de zgomot aditiv pe linia de măsurare.
Se formulează ipoteza că procesul dinamic asociat este de tip $PT2$ supra-amortizat.
Folosind rezultatele de la problema 1.5, se cere să se estimeze valorile parametrilor funcției de transfer ai procesului dinamic.

Tabela 1.3: Valorile variabilei timp în exempul din problema 1.6.

0.000	0.010	0.020	0.031	0.041	0.051	0.061	0.071	0.082	0.092	0.102	0.112	0.122
0.133	0.143	0.153	0.163	0.173	0.184	0.194	0.204	0.214	0.224	0.235	0.245	0.255
0.265	0.276	0.286	0.296	0.306	0.316	0.327	0.337	0.347	0.357	0.367	0.378	0.388
0.398	0.408	0.418	0.429	0.439	0.449	0.459	0.469	0.480	0.490	0.500	0.510	0.520
0.531	0.541	0.551	0.561	0.571	0.582	0.592	0.602	0.612	0.622	0.633	0.643	0.653
0.663	0.673	0.684	0.694	0.704	0.714	0.724	0.735	0.745	0.755	0.765	0.776	0.786
0.796	0.806	0.816	0.827	0.837	0.847	0.857	0.867	0.878	0.888	0.898	0.908	0.918
0.929	0.939	0.949	0.959	0.969	0.980	0.990	1.000					

Tabela 1.4: Valorile funcției pondere în exempul din problema 1.6.

-0.000	4.495	7.937	10.531	12.445	13.813	14.746	15.334	15.649	15.749	15.682	15.486	
15.192	14.824	14.404	13.945	13.462	12.964	12.459	11.954	11.452	10.959	10.475		
10.005	9.548	9.107	8.681	8.272	7.878	7.501	7.140	6.795	6.465	6.150	5.849	5.563
5.290	5.030	4.782	4.546	4.321	4.108	3.904	3.711	3.527	3.352	3.186	3.028	2.878
2.735	2.599	2.470	2.347	2.230	2.120	2.014	1.914	1.819	1.728	1.643	1.561	1.483
1.409	1.339	1.273	1.209	1.149	1.092	1.038	0.986	0.937	0.891	0.846	0.804	0.764
0.726	0.690	0.656	0.623	0.592	0.563	0.535	0.508	0.483	0.459	0.436	0.414	0.394
0.374	0.355	0.338	0.321	0.305	0.290	0.275	0.262	0.249	0.236	0.225		

Problema 1.7

Un proces dinamic proporțional cu întârziere de ordinul doi sub-amortizat, $PT2$ este reprezentat prin funcția de transfer:

$$G(s) = \frac{K_a \cdot \omega_n^2}{s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2},$$

în care K_a este constanta de proporționalitate ζ este factorul de amortizare și ω_n este pulsația naturală.
Se cer următoarele:

(a) să se arate că expresia funcției de transfer poate fi scrisă sub forma echivalentă:

$$G(s) = \frac{K_a \cdot (\omega_d^2 + \sigma_a^2)}{s^2 + 2\sigma_a \cdot s + \omega_d^2 + \sigma_a^2},$$

în care parametrii $\sigma_a = \zeta \cdot \omega_n$ reprezintă atenuarea și $\omega_d = \omega_n\sqrt{1-\zeta^2}$ este pulsația naturală amortizată ai procesului dinamic.

(b) să se calculeze expresia funcției pondere $g(t)$ a procesului dinamic;

(c) să se calculeze expresia funcției indiciale $g_1(t)$ a procesului dinamic.

Problema 1.8

Se cere să se demonstreze următoarele proprietăți ale funcțiilor pondere și indicială ale proceselor dinamice proporționale cu întârziere de ordinul doi sub-amortizate:

(a) Aria suprafeței cuprinsă între graficul funcției pondere și axa orizontală este numeric egală cu factorul de proporționalitate al funcției de transfer.

(b) Valoarea primului maxim al funcției pondere este dată de expresia:

$$g_{max,1} = \frac{K_a \left(\omega_d^2 + \sigma_a^2\right) e^{-\frac{\pi\sigma_a}{2\omega_d}}}{\omega_d}.$$

(c) Raportul dintre primele două maxime ale funcției pondere depinde de factorul de amortizare și nu depinde nici de pulsația naturală nici de factorul de proporționalitate.

(d) Valoarea la infinit a funcției indiciale este numeric egală cu factorul de proporționalitate al procesului dinamic.

Răspunsuri

Problema 1.1:

(a) $g(t) = \frac{K_a e^{-\frac{t}{T_f}}}{T_f}\chi(t)$, **(b)** $g_1(t) = K_a \cdot \left[1 - e^{\frac{-t}{T_f}}\right] \cdot \chi(t)$, **(c)** $g_1(t) = \int\limits_0^t g(\vartheta)d\vartheta$.

Problema 1.2: -

Problema 1.3: $\hat{K_a} = 5.001$, $\hat{T_f} = 0.200$.

Problema 1.4:

(a) $g(t) = \frac{K_a}{T_{f1}-T_{f2}}\left(e^{-\frac{t}{T_{f1}}} - e^{-\frac{t}{T_{f2}}}\right)\chi(t)$,
(b) $g_1(t) = K_a\left[1 - \frac{T_{f1}}{T_{f1}-T_{f2}}e^{-\frac{t}{T_{f1}}} + \frac{T_{f2}}{T_{f1}-T_{f2}}e^{-\frac{t}{T_{f2}}}\right]\chi(t)$,
(c) $g_1(t) = \int\limits_0^t g(\vartheta)d\vartheta$.

Problema 1.5:

(b) $\frac{T_{f1}T_{f2}\left(\log\left(T_{f1}\right)-\log\left(T_{f2}\right)\right)}{T_{f1}-T_{f2}}$; deci, nu depinde de K_a.

Problema 1.6: $\hat{K_a} = 4.952$, $\hat{T_f}1 = 0.049$, $\hat{T_f}2 = 0.196$.

Problema 1.7:

(b) $g(t) = \frac{K_a\left(\omega_d^2+\sigma_a^2\right)e^{-\sigma_a t}\sin\left(\omega_d t\right)\chi(t)}{\omega_d} = \frac{K_a\cdot\omega_n^2\cdot e^{-\sigma_a t}\sin\left(\omega_d t\right)\chi(t)}{\omega_d}$.

(c) $g_1(t) = \frac{K_a\left(\omega_d e^{\sigma_a t} - \omega_d\cos\left(\omega_d t\right) - \sigma_a\sin\left(\omega_d t\right)\right)e^{-\sigma_a t}\chi(t)}{\omega_d}$
sau $g_1(t) = K_a\left\{1 - e^{-\sigma_a t}\left[\cos(\omega_d t) + \frac{\sigma_a}{\omega_d}\sin(\omega_d t)\right]\right\}\chi(t)$.

Problema 1.8:

(c) $\frac{g_{max,1}}{g_{max,2}} = e^{\frac{2\pi\zeta}{\sqrt{1-\zeta^2}}}$.

Rezolvări

Rezolvarea problemei 1.1

Etapele rezolvării.

(a) - Calculul expresiei funcției pondere a procesului dinamic.

Funcția pondere reprezintă răspunsul sistemului dinamic la semnal de intrare impuls Dirac, $\delta(t)$.

Calculul funcției pondere se face cu transformarea Laplace inversă aplicată funcției de transfer, deci se folosește relația:

$$g(t) = \mathcal{L}^{-1}G(s).$$

(b) - Calculul expresiei funcției indiciale a procesului dinamic.

Funcția indicială reprezintă răspunsul sistemului dinamic la semnal de intrare treaptă unitară (funcția lui Heaviside notată cu $\chi(t)$).

Pentru calculul funcției indiciale, $g_1(t)$, mai întâi se calculează transformata Laplace a acesteia cu relația:

$$Y(s) = G(s) \cdot U(s) = G(s) \cdot \frac{1}{s},$$

în care $Y(s) = \mathcal{L}[g_1(t)]$, $G(s)$ este expresia funcției de transfer și $U(s) = \mathcal{L}[\chi(t)] = \frac{1}{s}$ este transformata Laplace a funcției treaptă unitară.

Expresia în domeniul timpului a funcției indiciale se calculează cu transformarea Laplace inversă aplicată funcției $Y(s)$:

$$g_1(t) = \mathcal{L}^{-1}[Y(s)].$$

(c) - Relația dintre funcția pondere și funcția indicială.

Funcțiile pondere și indicială reprezintă răspunsurile sistemului dinamic la semnale de intrare impuls Dirac și respectiv treaptă unitară; impulsul Dirac este derivata (în sens generalizat) a funcției treaptă unitară. Analog, funcția pondere este numeric egală cu derivata funcției indiciale.

Prin urmare, relația dintre funcția indicială și funcția pondere este:

$$g_1(t) = \int_0^t g(\vartheta)d\vartheta.$$

Relația de mai sus se verifică prin calcul direct.

Coduri - Aplicația 1.1.

Se apelează modulele necesare.

```
import sympy
import numpy
import scipy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Definițiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
g=sympy.Function('g')(t)#definitia functiei pondere a sistemului
    dinamic
u=sympy.Function('u')(t)#definitia functiei semnal la portul de
    intrare
y=sympy.Function('y')(t)#definitia functiei semnal la portul de
    iesire
Gs=Function('Gs')#definitia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
T_f=sympy.symbols('T_f',positive=True)#definitia constantei de timp
     T_f
Us=sympy.Function('Us')(s)#definitia transformatei Laplace a
    functiei u
Ys=sympy.Function('Ys')(s)#definitia transformatei Laplace a
    functiei y
```

```
Gs=K_a/(T_f*s+1)#expresia functiei de transfer
g=sympy.inverse_laplace_transform(Gs,s,t)#functia pondere se
    calculeaza cu transformarea Laplace inversa
Gs,g
```

Rezultat: $G(s) = \frac{K_a}{T_f s+1}$, $g(t) = \frac{K_a e^{-\frac{t}{T_f}} \chi(t)}{T_f}$.

Calculul expresiei funcției indiciale.

```
u=sympy.Heaviside(t)#expresia functiei semnal la portul de intrare
    - treapta unitara
Us=sympy.laplace_transform(u,t,s)[0]#transformata Laplace a
    functiei u
Ys=Gs*Us#transformata Laplace a functiei y - raspunsul sistemului
    dinamic
y=sympy.inverse_laplace_transform(Ys,s,t)#expresia in domeniul
    timpului a functiei y
y
```

Rezultat: $g_1(t) = K_a \chi(t) - K_a e^{-\frac{t}{T_f}} \chi(t)$.

Verificarea relației de legătură dintre funcția indicială și funcția pondere.

```
theta=sympy.symbols('theta',positive=True)#definitia variabilei
    auxiliare theta
g_theta=g.subs(t,theta)
g1=sympy.integrate(g_theta,(theta,0,t))#calculul integralei
    definite a raspunsului pondere
g1
```

Rezultat: $K_a - K_a e^{-\frac{t}{T_f}}$.

Reprezentare grafică.

```
dKa=5
dTf=0.2
dG=g.subs(T_f,dTf).subs(K_a,dKa)
dG1=y.subs(T_f,dTf).subs(K_a,dKa)
p1=plot(dG, (t,-10,10), label='g', line_color='green', xlabel='t',
    ylabel='g(t),g1(t)',
        show=False)
p2=plot(dG1, (t,-10,10), label='g1', line_color='blue',
        show=False)
p1.extend(p2)
p1.show()
```

Rezultat:

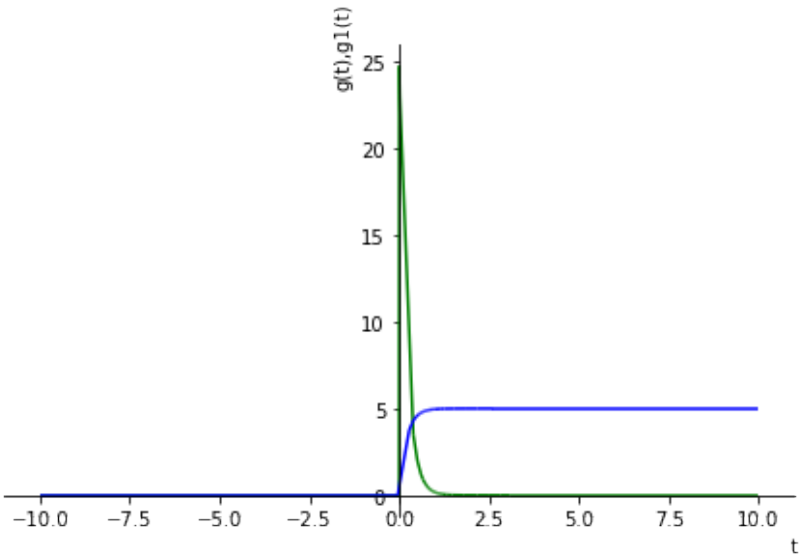


Figura 1.1: Reprezentarea grafică a funcțiilor pondere (verde) și indicială (albastru) ale unui proces $PT1$ cu parametrii $K_a = 5$ și $T_f = 0.2$.

Rezolvarea problemei 1.2

Etapele rezolvării.

Proprietatea (a)

Valoarea în origine a funcției pondere se obține prin calculul limitei:

$$g(0_+) = \lim_{t \to 0} \frac{K_a}{T_f} \cdot e^{\frac{-t}{T_f}}.$$

Proprietatea (b)

- se introduce notația $t_2 = t_1 + \Delta t$ și se scriu expresiile funcției pondere la cele două momente de timp:

$$g(t_1) = \frac{K_a}{T_f} e^{\frac{-t_1}{T_f}} \text{ și } g(t_2) = \frac{K_a}{T_f} e^{\frac{-(t_1+\Delta t)}{T_f}}$$

- apoi se rezolvă ecuația:

$$\frac{g(t_1)}{g(t_2)} - e = 0$$

în raport cu necunoscuta Δt.

Proprietatea (c)

- se calculează panta tangentei la graficul funcției pondere în punctul de abscisă 0_+:

$$m_0 = \left. \frac{dg(t)}{dt} \right|_{t \to 0_+},$$

- se calculează ordonata la origine a tangentei la graficul funcției pondere în punctul de abscisă 0_+:

$$n_0 = \lim_{t \to 0_+} g(t),$$

- se scrie ecuația dreptei tangente la grafic $(d) : y = m_0 \cdot t + n_0$.
  Abscisa punctului de intersecție dintre dreapta (d) și axa orizontală este soluția ecuației:

$$m_0 \cdot t + n_0 = 0.$$

Proprietatea (d)

Se calculează integrala:

$$A = \int_0^\infty g(t)dt.$$

Proprietatea (e)

- expresia funcției indiciale a fost calculată la punctul (b) de la problema 1.1.
- pentru verificarea proprietății (e), se calculează $\lim_{t \to \infty} g_1(t)$.

Coduri - Aplicația 1.2.

Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definitiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
g=sympy.Function('g')(t)#definitia functiei pondere a sistemului
    dinamic
u=sympy.Function('u')(t)#definitia functiei semnal la portul de
    intrare
y=sympy.Function('y')(t)#definitia functiei semnal la portul de
    iesire
Gs=Function('Gs')#definitia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
T_f=sympy.symbols('T_f',positive=True)#definitia constantei de timp
     T_f
Us=sympy.Function('Us')(s)#definitia transformatei Laplace a
    functiei u
Ys=sympy.Function('Ys')(s)#definitia transformatei Laplace a
    functiei y
```

Calculul expresiei funcției pondere.

```
Gs=K_a/(T_f*s+1)#expresia functiei de transfer
g=sympy.inverse_laplace_transform(Gs,s,t)#functia pondere se
    calculeaza cu transformarea Laplace inversa
sympy.print_latex(g)
Gs,g
```

Rezultat: $\frac{K_a e^{-\frac{t}{T_f}} \chi(t)}{T_f}$.

Verificarea proprietății (a).

```
g0=sympy.limit(g,t,0)
g0
```

Rezultat: $\frac{K_a}{T_f}$.

Verificarea proprietății (b).

```
t_1=sympy.symbols('t_1',positive=True)#definitia variabilei t_1 -
    momentul de timp t_1
det=sympy.symbols('det', positive=True)#defintia variabilei det -
    diferenta dintre momentele de timp t_1 si t_2
t_2=t_1+det#momentul de timp t_2
g_t1=g.subs(t,t_1)#expresia functiei pondere la momentul t_1
g_t2=g.subs(t,t_2)#expresia functiei pondere la momentul t_2
sympy.print_latex(g_t1)
```

```
sympy.print_latex(g_t2)
g_t1, g_t2
```

Rezultat: $\frac{K_a e^{-\frac{t_1}{T_f}}}{T_f}$ și $\frac{K_a e^{-\frac{det+t_1}{T_f}}}{T_f}$.

```
sympy.solve(g_t1/g_t2-sympy.exp(1), det)[0]#rezolvarea ecuatiei
    g_t1/g_t2-e = 0
```

Rezultat: T_f.

Verificarea proprietății (c).

```
g_plus=g.args[0]*g.args[1]*g.args[3]#restrictia functiei pondere in
    intervalul (0, +oo)
m=sympy.diff(g_plus,t)#panta tangentei la graficul functiei pondere
m0=m.subs(t,0)#panta tangentei la graficul functiei pondere in
    punctul de abscisa 0+
n0=g.subs(t,0)#ordonata punctului pe graficul functiei pondere in
    punctul de abscisa 0+
d=m0*t+n0#ecuatia dreptei tangenta la graficul functiei pondere in
    punctul de abscisa 0+
sympy.solve(d,t)[0]#calcul abscisei punctului de intersectie a
    dreptei cu axa orizontala
```

Rezultat: T_f.

Verificarea proprietății (d).

```
aria=sympy.integrate(g_plus,(t,0,oo))
aria
```

Rezultat: K_a.

Verificarea proprietății (e).

```
Us=sympy.laplace_transform(Heaviside(t),t,s)[0]#transformata
    Laplace a functiei treapta unitara
Ys=Gs*Us#transformata Laplace a functiei indiciale
g1=sympy.inverse_laplace_transform(Ys,s,t)#calculeaza expresia
    functiei indiciale
sympy.print_latex(g1)
g1
```

Rezultat: $K_a\chi(t) - K_a e^{-\frac{t}{T_f}}\chi(t)$.

```
sympy.limit(g1,t,oo)#calculeaza valoarea la infinit a functiei
    indiciale
```

Rezultat: K_a.

Rezolvarea problemei 1.3

Etapele rezolvării.

- Factorul de proporționalitate $\hat{K}_a$ se estimează pe baza proprietății (d).
  Se calculează integrala numerică a valorilor secvenței de eșantioane; punctul de la infinit se aproximează cu ultimul eșantion din secvență.

- Constanta de timp se calculează cu proprietatea (a) Primul eșantion din secvență aproximează valoarea funcției pondere în punctul de abscisă 0_+. $\hat{T}_f$ rezultă din expresia:

$$g(0_+) = \frac{K_a}{T_f} \Rightarrow \hat{T}_f = \frac{\hat{K}_a}{g(0_+)}.$$

Observație inportantă.
În acest exemplu, rezultatele estimării sunt apropiate de valorile adevărate ale parametrilor deoarece eșantioanele secvențelor nu sunt perturbate de zgomot pe linia de măsurare.
Dacă eșantioanele secvențelor ar fi perturbate de zgomot aditiv, atunci metoda prezentată ar furniza estimări cu abateri mari deoarece în calcule intervine doar valoarea unui anumit eșantion - perturbat cu zgomot (proces aleator) - la un anumit moment de timp.
De aceea, în practica identificării sunt implementate metode în cadrul cărora valorile tuturor eșantioanelor din secvențe sunt mediate/integrate pe ansamblu pentru a furniza estimările dorite ale parametrilor.

Coduri - Aplicația 1.3.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aTimp=numpy.loadtxt('Cap1Prob13_aTimp.txt')#preia datele din fisier
    - secventa la portul de intrare
aY=numpy.loadtxt('Cap1Prob13_aY.txt')#preia datele din fisier -
   secventa la portul de iesire
```

Reprezentarea grafică a datelor prelevate.

```
fig, ax=pyplot.subplots(nrows=1, ncols=1)
ax.scatter(aTimp,aY, label='Valori masurate')
ax.plot(aTimp,aY, label='Functia pondere')
ax.grid(True)
ax.set_xlabel('Timpul')
ax.set_ylabel('Functia pondere')
ax.legend()
```

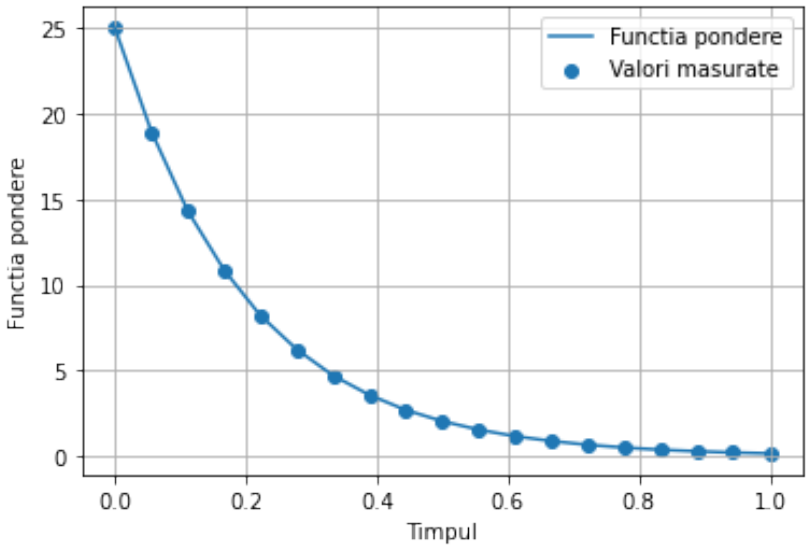


Figura 1.2: Valorile funcției pondere în experimentul din problema 1.3.

Rezultat: în figura (1.2).

Estimarea parametrilor funcției de transfer.

```
dKa=numpy.trapz(aY,aTimp)#integrare numerica cu metoda trapezelor
dKa,format(dKa,'5.3f')
```

Rezultat: $\hat{K}_a = 5.001$.

```
dGmax=numpy.max(aY)#preia valoarea maxima a functiei pondere - G0
dTf=dKa/dGmax#estimarea constantei de timp cu relatia Ka/Tf=g0
dTf, format(dTf,'5.3f')
```

Rezultat: $\hat{T}_f = 0.200$.

Rezolvarea problemei 1.4

Etapele rezolvării. **(a) - Calculul expresiei funcției pondere a procesului dinamic.**
Funcția pondere se calculează cu transformarea Laplace inversă aplicată funcției de transfer.

$$g(t) = \mathcal{L}^{-1}G(s).$$

(b) - Calculul expresiei funcției indiciale a procesului dinamic.
Funcția indicială se calculează cu relațiile:

$$Y(s) = G(s) \cdot U(s) = G(s) \cdot \frac{1}{s},$$

în care $Y(s) = \mathcal{L}[g_1(t)]$, $G(s)$ este expresia funcției de transfer și $U(s) = \mathcal{L}[\chi(t)] = \frac{1}{s}$ este transformata Laplace a funcției treaptă unitară.
Expresia în domeniul timpului a funcției indiciale se calculează cu transformarea Laplace inversă aplicată funcției $Y(s)$:

$$g_1(t) = \mathcal{L}^{-1}[Y(s)].$$

(c) - Relația dintre funcția pondere și funcția indicială.
Funcția indicială este numeric egală cu integrala funcției funcției pondere.

$$g_1(t) = \int_0^t g(\vartheta)d\vartheta.$$

Coduri - Aplicația 1.4.
Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
g=sympy.Function('g')(t)#definitia functiei pondere a sistemului
    dinamic
u=sympy.Function('u')(t)#definitia functiei semnal la portul de
    intrare
y=sympy.Function('y')(t)#definitia functiei semnal la portul de
    iesire
Gs=Function('Gs')#definitia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
T_f1=sympy.symbols('T_f1',positive=True)#definitia constantei de
    timp T_f
T_f2=sympy.symbols('T_f2',positive=True)#definitia constantei de
    timp T_f
```

```
Us=sympy.Function('Us')(s)#definitia transformatei Laplace a
    functiei u
Ys=sympy.Function('Ys')(s)#definitia transformatei Laplace a
    functiei y
```

Calculul expresiei funcției pondere.

```
Gs=K_a/((T_f1*s+1)*(T_f2*s+1))#expresia functiei de transfer
g=sympy.inverse_laplace_transform(Gs,s,t)#functia pondere se
    calculeaza cu transformarea Laplace inversa
Gs,g
```

Rezultat: $g(t) = -\frac{K_a e^{-\frac{t}{T_{f2}}}\chi(t)}{T_{f1}-T_{f2}} + \frac{K_a e^{-\frac{t}{T_{f1}}}\chi(t)}{T_{f1}-T_{f2}}$.

Expresia de mai sus poate fi rescrisă și sub forma:

$$g(t) = \frac{K_a}{T_{f1}-T_{f2}}\left(e^{-\frac{t}{T_{f1}}} - e^{-\frac{t}{T_{f2}}}\right)\chi(t).$$

Calculul expresiei funcției indiciale.

```
u=sympy.Heaviside(t)#expresia functiei semnal la portul de intrare
    - treapta unitara
Us=sympy.laplace_transform(u,t,s)[0]#transformata Laplace a
    functiei u
Ys=Gs*Us#transformata Laplace a functiei y - raspunsul sistemului
    dinamic
y=sympy.inverse_laplace_transform(Ys,s,t)#expresia in domeniul
    timpului a functiei y
sympy.factor(y)
```

Rezultat: $g_1(t) = \frac{K_a T_{f1}\theta(t)}{T_{f1}-T_{f2}} - \frac{K_a T_{f1} e^{-\frac{t}{T_{f1}}}\theta(t)}{T_{f1}-T_{f2}} - \frac{K_a T_{f2}\theta(t)}{T_{f1}-T_{f2}} + \frac{K_a T_{f2} e^{-\frac{t}{T_{f2}}}\theta(t)}{T_{f1}-T_{f2}}$.

Expresia de mai sus poate fi rescrisă și sub forma:

$$g_1(t) = K_a\left[1 - \frac{T_{f1}}{T_{f1}-T_{f2}}e^{-\frac{t}{T_{f1}}} + \frac{T_{f2}}{T_{f1}-T_{f2}}e^{-\frac{t}{T_{f2}}}\right]\chi(t).$$

Verificarea identității de la punctul (c).

```
theta=sympy.symbols('theta',positive=True)#definitia variabilei
    auxiliare theta
g_theta=g.subs(t,theta)
g1_int=sympy.integrate(g_theta,(theta,0,t))#calculul integralei
    definite a raspunsului pondere
g1_int
```

Rezultat: $g_{1int} = \frac{K_a T_{f1}}{T_{f1}-T_{f2}} - \frac{K_a T_{f1} e^{-\frac{t}{T_{f1}}}}{T_{f1}-T_{f2}} - \frac{K_a T_{f2}}{T_{f1}-T_{f2}} + \frac{K_a T_{f2} e^{-\frac{t}{T_{f2}}}}{T_{f1}-T_{f2}}$.

```
g1-g1_int#verificarea identitatii rezultatelor
```

Rezultat: 0. Deci, cele două expresii sunt identice.

Reprezentare grafică - exemplu.

```
dKa=5
dTf1=0.2
dTf2=0.05
dG=g.subs(T_f1,dTf1).subs(T_f2,dTf2).subs(K_a,dKa)
dG1=y.subs(T_f1,dTf1).subs(T_f2,dTf2).subs(K_a,dKa)
p1=plot(dG, (t,-1,1), label='g', line_color='green', xlabel='t',
    ylabel='g(t),g1(t)',
          show=False)
p2=plot(dG1, (t,-1,1), line_color='blue',
          show=False)
p1.extend(p2)
p1.show()
```

Rezultat:

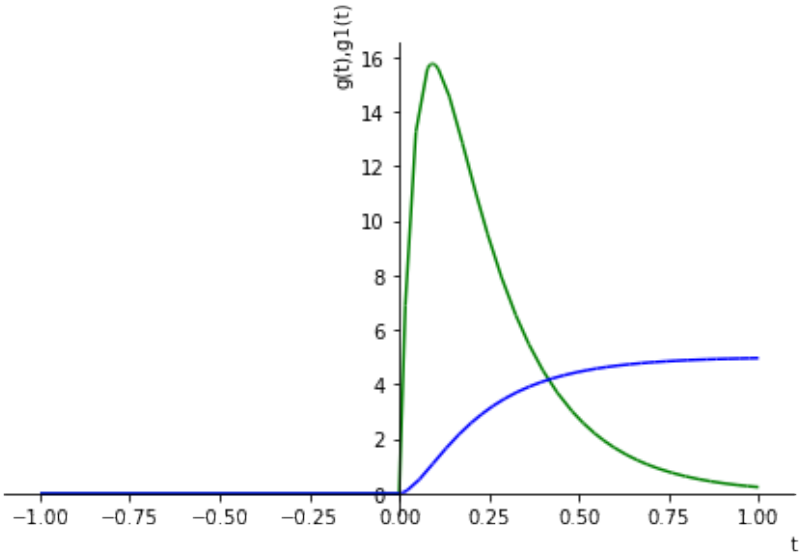


Figura 1.3: Reprezentarea grafică a funcțiilor pondere și indicială pentru un proces $PT2$ supra-amortizat. Valorile parametrilor procesului sunt $K_a = 5$, $T_{f1} = 0.2$ și $T_{f2} = 0.05$.

Rezolvarea problemei 1.5

Etapele rezolvării.
Proprietatea (a). Se calculează integrala:

$$S = \int_0^\infty g(t)dt \Rightarrow S = K_a.$$

Proprietatea (b). Abscisa punctului de maxim al funcției pondere se obține cu teorema lui Fermat. Abscisa punctului este soluția ecuației:

$$\frac{dg(t)}{dt} = 0.$$

Proprietatea (c). Se calculează limita $\lim_{t\to\infty} g_1(t)$.
Proprietatea (d). Se verifică prin evaluarea expresiei din enunțul problemei.

Coduri - Aplicația 1.5.
Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
g=sympy.Function('g')(t)#definitia functiei pondere a sistemului
    dinamic
u=sympy.Function('u')(t)#definitia functiei semnal la portul de
    intrare
y=sympy.Function('y')(t)#definitia functiei semnal la portul de
    iesire
Gs=Function('Gs')#definitia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
T_f1=sympy.symbols('T_f1',positive=True)#definitia constantei de
    timp T_f
T_f2=sympy.symbols('T_f2',positive=True)#definitia constantei de
    timp T_f
Us=sympy.Function('Us')(s)#definitia transformatei Laplace a
    functiei u
Ys=sympy.Function('Ys')(s)#definitia transformatei Laplace a
    functiei y
```

Calculul expresiei funcției pondere.

```
Gs=K_a/((T_f1*s+1)*(T_f2*s+1))#expresia functiei de transfer
g=sympy.inverse_laplace_transform(Gs,s,t)#functia pondere se
    calculeaza cu transformarea Laplace inversa
Gs,g
```

$g(t) = -\frac{K_a e^{-\frac{t}{T_{f2}}}\theta(t)}{T_{f1}-T_{f2}} + \frac{K_a e^{-\frac{t}{T_{f1}}}\theta(t)}{T_{f1}-T_{f2}}$.

Calculul expresiei funcției indiciale.

```
u=sympy.Heaviside(t)#expresia functiei semnal la portul de intrare
    - treapta unitara
Us=sympy.laplace_transform(u,t,s)[0]#transformata Laplace a
    functiei u
Ys=Gs*Us#transformata Laplace a functiei y - raspunsul sistemului
    dinamic
y=sympy.inverse_laplace_transform(Ys,s,t)#expresia in domeniul
    timpului a functiei y - functia indiciala
sympy.factor(y)
```

Rezultat: $\frac{K_a T_{f1}\theta(t)}{T_{f1}-T_{f2}} - \frac{K_a T_{f1} e^{-\frac{t}{T_{f1}}}\theta(t)}{T_{f1}-T_{f2}} - \frac{K_a T_{f2}\theta(t)}{T_{f1}-T_{f2}} + \frac{K_a T_{f2} e^{-\frac{t}{T_{f2}}}\theta(t)}{T_{f1}-T_{f2}}$.

Verificarea proprietății (a).

```
aria=sympy.integrate(g,(t,0,oo))#calculul ariei suprafetei dintre
    graficul functiei pondere si axa orizontala
sympy.simplify(aria)
```

Rezultat: K_a.

Verificarea proprietății (b).

```
#restrictia functiei pondere in intervalul (0,oo)
g_plus=g.args[0].args[0]*g.args[0].args[1]*g.args[0].args[3]-g.args
    [1].args[1]*g.args[1].args[2]*g.args[1].args[4]
g_deri=sympy.diff(g_plus,t)#calculul derivatei functiei pondere
t_deri=sympy.solve(g_deri,t)#calculul abscisei punctului de maxim
    functiei pondere
t_der=sympy.expand(t_deri[0])#expresia abscisei punctului de maxim
    al functiei pondere
sympy.factor(t_der)
```

Rezultat: $\frac{T_{f1}T_{f2}\log\left(T_{f1}\right)}{T_{f1}-T_{f2}} - \frac{T_{f1}T_{f2}\log\left(T_{f2}\right)}{T_{f1}-T_{f2}}$.

Verificarea proprietății (c).

```
sympy.limit(y,t,oo)#calculul limitei la infinit a functiei
    indiciale
```

Rezultat: K_a.

Verificarea proprietății (d).

```
tau=sympy.symbols('tau', positive=True)#definitia variabilei
    simbolice tau
a1=sympy.symbols('a1',positive=True)#definitia variabilei a1
a2=sympy.symbols('a2',positive=True)#definitia variabilei a2
g_plus_tau=g_plus.subs(t,t+tau)#expresia functiei pondere deplasata
     cu tau
```

```
g_plus_2tau=g_plus.subs(t,t+2*tau)#expresia functiei pondere
    deplasata cu 2.tau
a1=sympy.exp(-tau/T_f1)#expresia variabilei a1
a2=sympy.exp(-tau/T_f2)#expresia variabilei a2
```

```
expr=g_plus_2tau-(a1+a2)*g_plus_tau+a1*a2*g_plus#calculul si
    verificarea expresiei din enuntul problemei
sympy.simplify(expr)
```

Rezultat: 0.

Rezolvarea problemei 1.6

Etapele rezolvării.

Calculul constantei de proporționalitate.

Se face pe baza proprietății (a) din problema precedentă prin integrarea numerică a valorilor din secvența funcției pondere.

Calcul constantelor de timp.

Se face pe baza proprietății (d) din problema precedentă după cum urmează.

- Se alege o valoare a deplasării în domeniul timpului $\tau = 5$ eșantioane și două valori inițiale ale momentelor de timp t_1 și t_2 din secvență astfel poziționate încât să includă intervalele de creștere și descreștere ale valorilor funcției pondere.
  De exemplu: prima valoare la momentul inițial și a doua valoare la 0.1 secunde după momentul inițial.

- Pe baza proprietății (d), se formează următorul sistem de ecuații, în care $S = a_1 + a_2$ și $P = a_1 \cdot a_2$:

$$\begin{cases} -S \cdot g(t_1 + \tau) + P \cdot g(t_1) &= -g(t_1 + 2\tau) \\ -S \cdot g(t_2 + \tau) + P \cdot g(t_2) &= -g(t_2 + 2\tau) \end{cases}.$$

- Sistemul de ecuații de mai sus se scrie sub formă matriceală și se rezolvă în raport cu necunoscutele S și P:

$$\begin{bmatrix} -g(t_1 + \tau) & g(t_1) \\ -g(t_2 + \tau) & g(t_2) \end{bmatrix} \times \begin{bmatrix} S \\ P \end{bmatrix} = \begin{bmatrix} -g(t_1 + 2\tau) \\ -g(t_2 + 2\tau) \end{bmatrix}.$$

- Pentru calculul valorilor necunoscutelor a_1 și a_2, se formează ecuația de gradul doi în S și P:

$$\xi^2 - S \cdot \xi + P = 0.$$

- Calculul constantelor de timp se face pornind de la relațiile de definiție:

$$a_1 = e^{\frac{-\tau}{T_{f1}}} \text{ și } a_2 = e^{\frac{-\tau}{T_{f2}}}.$$

Coduri - Aplicația 1.6.

Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aTimp=numpy.loadtxt('Cap1Prob16_aTimp.txt')#preia datele din fisier
    - secventa la portul de intrare
aY=numpy.loadtxt('Cap1Prob16_aY.txt')#preia datele din fisier -
   secventa la portul de iesire
```

Reprezentarea grafică a datelor prelevate.

```
fig , ax=pyplot.subplots(nrows=1, ncols=1)
ax.scatter(aTimp,aY, label='Valori masurate', marker='.')
ax.plot(aTimp,aY, label='Functia pondere')
ax.grid(True)
ax.set_xlabel('Timpul')
ax.set_ylabel('Functia pondere')
ax.legend()
```

Rezultat:

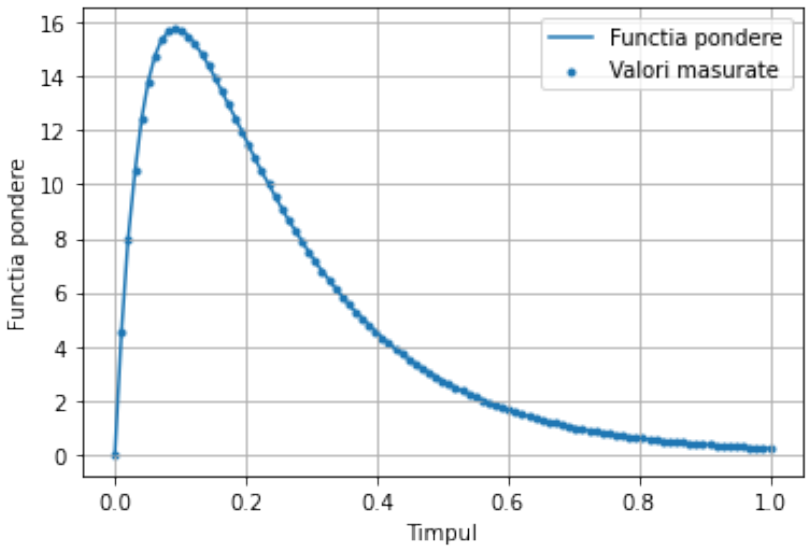


Figura 1.4: Eșantioanele funcției pondere în experimentul din problema 1.6.

Estimarea valorilor funcției de transfer.

```
dKa=numpy.trapz(aY,aTimp)#integrare numerica cu metoda trapezelor ;
    valoarea adevarata = 5
dKa,format(dKa,'5.3f')
```

Rezultat: $\hat{K}_a = 4.952$.

Definirea momentelor de timp la care se fac calculele.

```
dTe=0.01#perioada de esantionare
iTau=5#indexul esantioanelor deplasarii in timp
iT1=0#indexul primului moment de timp
iT2=iT1+iTau#indexul celui de-al doilea moment de timp
iT3=iT1+2*iTau#indexul celui de-al treilea moment de timp
iT4=10#indexul celui de-al patrulea moment de timp
iT5=iT4+iTau#indexul celui de-al cincilea moment de timp
iT6=iT4+2*iTau#indexul celui de-al saselea moment de timp
```

Definirea elementelor matricelor sistemului de ecuații liniare.

```
aA=numpy.matrix([[-aY[iT2], aY[iT1]],[-aY[iT5],aY[iT4]]])
aB=numpy.matrix([[-aY[iT3]],[-aY[iT6]]])
aA,aB
```

Rezultat:
$aA = ([[-13.813, -0.], [-13.945, 15.682]])$, $aB = ([[-15.682], [-11.452]])$

Rezolvarea sistemului de ecuații liniare.

```
aX=numpy.linalg.solve(aA,aB)
aX
```

Rezultat:
$aX = [[1.13530732], [0.27929222]]$

Rezolvarea ecuației polinomiale de gradul doi în S și P.

```
dS=aX[0,0]#valoarea sumei necunoscutelor S=a1+a2
dP=aX[1,0]#valoarea produsului necunoscutelor P=a1.a2
xi=sympy.symbols('xi', real=True)#definitia variabilei simbolice xi
sol=sympy.solve(xi**2-dS*xi+dP,xi)#calculul solutiei ecuatiei de
    gradul doi
dA1=sol[0]#valoarea necunoscutei a1
dA2=sol[1]#valoarea necunoscutei a2
dA1,dA2, format(dA1,'5.3f'), format(dA2,'5.3f')
```

Rezultat: $a_1 = 0.360$ și $a_2 = 0.775$.

Calculul valorilor constantelor de timp.

```
Tf1=sympy.symbols('Tf1', real=True)#definitia variabilei simbolice
    Tf1
Tf2=sympy.symbols('Tf2', real=True)#definitia variabilei simbolice
    Tf2
dTe=0.01#perioada de esantionare
dTau=iTau*dTe#deplasarea in domeniul timpului
dTf1=sympy.solve(sympy.exp(-dTau/Tf1)-dA1,Tf1)#a1=exp(-tau/Tf1) ;
    valoarea adevarata = 0.05
dTf2=sympy.solve(sympy.exp(-dTau/Tf1)-dA2,Tf1)#a1=exp(-tau/Tf2) ;
    valoarea adevarata = 0.2
dTf1[0],dTf2[0]
```

```
format(dTf1[0],'5.3f'), format(dTf2[0],'5.3f')
```

Rezultat: $T_{f1} = 0.049$ și $T_{f2} = 0.196$.

Observații

- Deși valorile eșantioanelor secvențelor nu sunt perturbate de zgomot aditiv pe liniile de măsurare, totuși rezultatele identificării nu coincid cu valorile adevărate - rezultatele sunt aproximări, foarte apropiate de valorile adevărate. Explicația este că identificarea s-a făcut pe un număr finit de puncte distincte de observare.

- Dacă eșantioanele secvențelor ar fi perturbate de zgomot aditiv, atunci metoda implementată ar furniza aproximări foarte diferite de valorile adevărate, deoarece estimarea se face doar cu șase valori distincte din secvențe, perturbate de valori particulare ale zgomotului (care este un proces aleator).

Rezolvarea problemei 1.7

Etapele rezolvării.
(a) Verificarea echivalenței delor două expresii ale funcției de transfer se face prin înlocuire și pe baza egalității $\omega_n^2 = \omega_d^2 + \sigma_a^2$.

(b) Funcția pondere numeric egală cu transformata Laplace inversă a funcției de transfer, $g(t) = \mathcal{L}^{-1}[G(s)]$.

(c) Calculul funcției indiciale se face astfel:

- se calculează transformata Laplace a funcției treaptă unitară, $U(s) = \mathcal{L}[\chi(t)]$.
- se calculează transformata Laplace a funcției indiciale cu formula, $Y(s) = G(s) \cdot U(s)$.
- cu transformarea Laplace inversă aplicată funcției $Y(s)$, se calculează expresia în domeniul timpului a funcției indiciale, $y(t) = \mathcal{L}^{-1}[Y(s)] = g_1(t)$.

Coduri - Aplicația 1.7.
Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
g=sympy.Function('g')(t)#definitia functiei pondere a sistemului
    dinamic
u=sympy.Function('u')(t)#definitia functiei semnal la portul de
    intrare
y=sympy.Function('y')(t)#definitia functiei semnal la portul de
    iesire
Gs=Function('Gs')(s)#definitia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
sigma_a=sympy.symbols('sigma_a',positive=True)#definitia
    amortizarii
zeta=sympy.symbols('zeta',positive=True)#definitia factorului de
    amortizare
omega_n=sympy.symbols('omega_n',positive=True)#definitia pulsatiei
    naturale
omega_d=sympy.symbols('omega_d',positive=True)#definitia pulsatiei
    naturale amortizate
Us=sympy.Function('Us')(s)#definitia transformatei Laplace a
    functiei u
Ys=sympy.Function('Ys')(s)#definitia transformatei Laplace a
    functiei y
```

Verificarea echivalenței expresiilor funcției de transfer.

```
s_1=-sigma_a-sympy.I*omega_d#polul s_1 al functiei de transfer
s_2=-sigma_a+sympy.I*omega_d#polul s_2 al functiei de transfer
Gs=K_a*(omega_d**2+sigma_a**2)/((s-s_1)*(s-s_2))#expresia functiei
    de transfer
Gs1=sympy.factor(sympy.expand(Gs))
Gs1
```

Rezultat: $G(s) = \frac{K_a\left(\omega_d^2+\sigma_a^2\right)}{\omega_d^2+s^2+2s\sigma_a+\sigma_a^2}$

```
sympy.simplify(Gs1.subs(omega_d,omega_n*sympy.sqrt(1-zeta**2)).subs
    (sigma_a,zeta*omega_n))
```

Rezultat: $G(s) = \frac{K_a\omega_n^2}{\omega_n^2+2\omega_n s\zeta+s^2}$.

Calculul expresiei funcției pondere.

```
g=sympy.inverse_laplace_transform(Gs,s,t)
g
```

Rezultat: $g(t) = \frac{K_a\left(\omega_d^2+\sigma_a^2\right)e^{-\sigma_a t}\sin(\omega_d t)\chi(t)}{\omega_d}$.

Calculul expresiei funcției indiciale.

```
u=sympy.Heaviside(t)#semnalul la portul de intrare - functia
    treapta unitara
Us=sympy.laplace_transform(u,t,s)[0]#calculul transformatei Laplace
    a functiei u
Ys=Gs*Us#calculul transformatei Laplace a semnalului la portul de
    iesire
g1=sympy.inverse_laplace_transform(Ys,s,t)#calculul transformatei
    Laplace inverse a functiei Ys
g1t=sympy.expand(sympy.simplify(g1))
g1t.factor(K_a)
```

Rezultat: $g_1(t) = \frac{K_a\left(\omega_d e^{\sigma_a t}-\omega_d\cos(\omega_d t)-\sigma_a\sin(\omega_d t)\right)e^{-\sigma_a t}\chi(t)}{\omega_d}$.

Reprezentare grafică - exemplu.

```
dKa=5#valoarea factorului de proportionalitate
dZeta=0.4#valoarea factorului de amortizare
dOmegaN=5#valoarea pulsatiei naturale
dSigmaA=dZeta*dOmegaN
dOmegaD=dOmegaN*sympy.sqrt(1-dZeta**2)
dSigmaA, dOmegaD, format(dSigmaA,'5.3f'),format(dOmegaD,'5.3f')
```

Rezultat: $\sigma_a = 2.000$ și $\omega_d = 4.583$.

```
dGt=g.subs(K_a,dKa).subs(sigma_a,dSigmaA).subs(omega_n,dOmegaN).
    subs(omega_d,dOmegaD)
dGt
```

```
dG1t=g1t.subs(K_a,dKa).subs(sigma_a,dSigmaA).subs(omega_d,dOmegaD).
    subs(omega_n,dOmegaN)
dG1t
```

```
p1=sympy.plot(dGt,(t,-3,3),line_color='blue',show=False,ylabel='g(t
    ), g1(t)')
p2=sympy.plot(dG1t,(t,-3,3),line_color='green',show=False)
p1.extend(p2)
p1.show()
```

Rezultat:

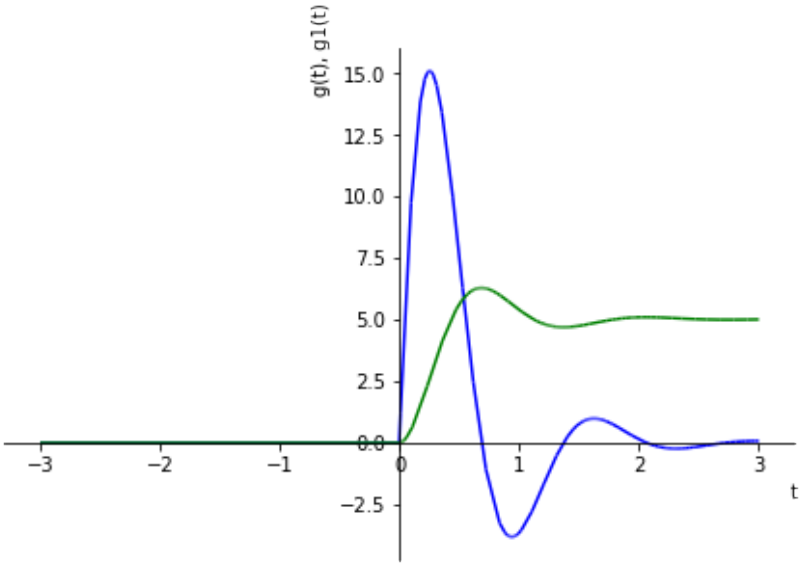


Figura 1.5: Reprezentările grafice ale funcțiilor pondere și indicială - exemplu. Parametrii funcției de transfer sunt: $K_a = 5$, $\zeta = 0.4$ și $\omega_n = 5$.

Rezolvarea problemei 1.8

Etapele rezolvării.

Proprietatea (a)

Aria suprafeței cuprinsă între graficul funcției pondere și axa orizontală se calculează cu formula:

$$S = \int_0^{\infty} g(t)dt \Rightarrow S = K_a.$$

Observație.

Rezultatul este similar cu cazul proceselor $PT1$ și $PT2$ supra-amortizat și permite estimarea factorului de proporționalitate al funcției de transfer din date prelevate prin măsurători.

Proprietatea (b)

- Se definește pseudo-perioada funcției pondere:

$$T_d = \frac{2\pi}{\omega_d}.$$

- Valoarea primului maxim al funcției pondere se obține la un sfert de pseudo-perioadă de la momentul inițial: $g_{max,1} = g\left(\frac{T_d}{4}\right)$.

Proprietatea (c)

Valoarea celui de-al doilea maxim al funcției pondere se obține la momentul $T_d + \frac{T_d}{4} = \frac{5}{4} \cdot T_d$ deci, $g_{max,2} = g\left(\frac{5 \cdot T_d}{4}\right)$ și se calculează raportul:

$$\frac{g_{max,1}}{g_{max,2}} = \frac{g\left(\frac{T_d}{4}\right)}{g\left(\frac{5 \cdot T_d}{4}\right)}.$$

Proprietatea (d)

Se calculează limita $\lim_{t \to \infty} g_1(t)$.

Coduri - Aplicația 1.8.

Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
g=sympy.Function('g')(t)#definitia functiei pondere a sistemului
    dinamic
u=sympy.Function('u')(t)#definitia functiei semnal la portul de
    intrare
y=sympy.Function('y')(t)#definitia functiei semnal la portul de
    iesire
```

```
Gs=Function('Gs')(s)#definitia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
sigma_a=sympy.symbols('sigma_a',positive=True)#definitia
    amortizarii
zeta=sympy.symbols('zeta',positive=True)#definitia factorului de
    amortizare
omega_n=sympy.symbols('omega_n',positive=True)#definitia pulsatiei
    naturale
omega_d=sympy.symbols('omega_d',positive=True)#definitia pulsatiei
    naturale amortizate
Us=sympy.Function('Us')(s)#definitia transformatei Laplace a
    functiei u
Ys=sympy.Function('Ys')(s)#definitia transformatei Laplace a
    functiei y
```

Calculul expresiei funcției pondere.

```
s_1=-sigma_a-sympy.I*omega_d#polul s_1 al functiei de transfer
s_2=-sigma_a+sympy.I*omega_d#polul s_2 al functiei de transfer
Gs=K_a*(omega_d**2+sigma_a**2)/((s-s_1)*(s-s_2))#expresia functiei
    de transfer
g=sympy.inverse_laplace_transform(Gs,s,t)
```

Calculul expresiei funcției indiciale.

```
u=sympy.Heaviside(t)#semnalul la portul de intrare - functia
    treapta unitara
Us=sympy.laplace_transform(u,t,s)[0]#calculul transformatei Laplace
     a functiei u
Ys=Gs*Us#calculul transformatei Laplace a semnalului la portul de
    iesire
g1=sympy.inverse_laplace_transform(Ys,s,t)#calculul transformatei
    Laplace inverse a functiei Ys
g1t=sympy.expand(sympy.simplify(g1))
g1t.factor(K_a)
```

Verificarea proprietății (a).

```
sympy.simplify(sympy.integrate(g,(t,0,oo)))
```

Rezultat: K_a.

Verificarea proprietății (b).

```
T_d=2*sympy.pi/omega_d#defintia pseudoperioadei procesului
g_max1=g.subs(t,T_d/4)#expresia primului maxim al functiei pondere
g_max1
```

Rezultat: $g_{max,1} = \frac{K_a\left(\omega_d^2+\sigma_a^2\right)e^{-\frac{\pi\sigma_a}{2\omega_d}}}{\omega_d}$.

```
sympy.simplify(g_max1.subs(sigma_a,zeta*omega_n).subs(omega_d,
    omega_n*sympy.sqrt(1-zeta**2)))#expresia primului
```

```
#maxim al functiei pondere in functie de factorul de amortizare si
    de pulsatia naturala
```

Rezultat: $\frac{K_a \omega_n e^{-\frac{\pi\zeta}{2\sqrt{1-\zeta^2}}}}{\sqrt{1-\zeta^2}}$.

Verificarea proprietății (c).

```
g_max2=g.subs(t,T_d/4+T_d)
raport=g_max1/g_max2#calculul raportului dintre primele doua maxime
    locale ale functiei pondere
raport.subs(sigma_a,zeta*omega_n).subs(omega_d,omega_n*sympy.sqrt
    (1-zeta**2))
```

Rezultat: $e^{\frac{2\pi\zeta}{\sqrt{1-\zeta^2}}}$; deci, depinde doar de ζ.

Verificarea proprietății (d).

```
sympy.limit(g1t,t,oo)#calculul limitei la infinit a functiei
    indiciale
```

Rezultat: K_a.

2

Variabile aleatoare și funcții de probabilitate

Enunțuri

Problema 2.1

Expresia funcției densitate de probabilitate asociată unei variable aleatoare cu distribuție normală (gaussiană), X este următoarea:

$$p(x) = \frac{1}{\sqrt{2 \cdot \pi} \cdot e^{\frac{-x^2}{2}}}.$$

Se cere să se calculeze:

(a) Valoarea mediei statistice, μ_x a variabilei aleatoare.

(b) Valoarea variației, σ_x^2 variabilei aleatoare.

(c) Probabilitatea ca valorile variabilei aleatoare să fie cuprinse în intervalul $(-3; +3)$.

Problema 2.2

Se dă funcția:

$$p(x) = \frac{c}{e^x + e^{-x}}, \ \ x \in \mathbb{R}.$$

(a) Să se determine valoarea constantei c astfel incât funcția $p(x)$ să reprezinte densitatea de repartiție a unei variabile aleatoare X.

(b) Dacă X si Y sunt două variable aleatoare independente statistic având densitatea de repartiție $p(\cdot)$, să se calculeze probabilitatea condiționată $P(X < 1, Y > 1)$.

Problema 2.3

Expresia funcției densitate de repartiție asociată unei variabile aleatoare X este următoarea:

$$p(x) = \begin{cases} \frac{2}{\pi \cdot (1+x^2)} & \text{dacă} \quad x \in (-1; 1); \\ 0 & \text{dacă} \quad x \in (-\infty; -1) \bigcup x \in [1; \infty). \end{cases}$$

Folosind definițiile momentelor statistice și variației, se cere să se calculeze:

(a) Să se verifice că funcția $p(x)$ din enunțul problemei poate reprezenta funcția densitate de probabilitate a unei variabile aleatoare, X.

(b) Valorile mediei statistice, μ_x și variației, σ_x^2 variabilei aleatoare X.

(c) Valoarea mediei statistice, μ_y a variabilei aleatoare $Y = X^2$.

Răspunsuri

Problema 2.1:

(a) $\mu_x = 0$; **(b)** $\sigma_x^2 = 0$; **(c)** 0.9973.

Problema 2.2:

(a) $c = \frac{2}{\pi}$, **(b)** $P(X < 1, Y > 1) = 0.174$.

Problema 2.3

(a) - ; **(b)** $\mu_x = 0$; $\sigma_x^2 = -1 + \frac{4}{\pi}$, **(c)** $\mu_{x^2} = -1 + \frac{4}{\pi}$.

Rezolvări

Problema 2.1

Etapele rezolvării.

(a) - Calculul valorii mediei statistice a variabilei aleatoare X

Se calculează integrala care urmează.

$$\mu_x = \int_{-\infty}^{+\infty} x \cdot p(x)dx = \int_{-\infty}^{+\infty} \frac{x}{\sqrt{2 \cdot \pi} \cdot e^{\frac{-x^2}{2}}} dx$$

(b) - Calculul valorii variației statistice a variabilei aleatoare X

Se calculează integrala care urmează.

$$\sigma_{x^2} = \int_{-\infty}^{+\infty} (x - \mu_x)^2 \cdot p(x)dx = \int_{-\infty}^{+\infty} [x - \mu_x]^2 \cdot \frac{1}{\sqrt{2 \cdot \pi} \cdot e^{\frac{-x^2}{2}}} dx$$

Coduri - Aplicația 2.1.

Se apelează modulele necesare.

```
import numpy#modul pentru calcule numerice
import scipy#modul pentru calcule stiintifice
import sympy#modul pentru calcul simbolic
from sympy import*#importa toata colectia de functii ale modulului
from matplotlib import pyplot#importa colectia de functii pentru
    reperzentare grafica
sympy.init_printing()#comada pentru initializarea printarii
```

Se definesc variabilele simbolice.

```
x=symbols('x', real=True)#definitia variabilei x
mu_x=symbols('mu_x', real=True)#definitia variabilei mu_x, media
    statistica rezultata prin calcul
var_x=symbols('var_x', real=True)#definitia variabilei var_x,
    varianta rezultata prin calcul
p=Function('p')(x)#defintia functiei p, densitatea de probabilitate
```

Se definește expresia funcției densitate de probabilitate.

```
p=1/(sqrt(2*pi))*exp(-x**2/2)#expresia functiei p - distributia de
    probabilitate
p#afiseaza rezultatul
```

Se verifica dacă funcția din enunț îndeplinește condiția să reprezinte o funcție densitate de probabilitate

```
integrate(p,(x,-oo,oo))#verificarea conditiei - reprezinta aria
    suprafetei marginita de axa orizontala si graficul functiei
```

Rezultat: 1

Se calculează valoarea mediei statistice, mu_x a distribuției.

```
mu_x=integrate(x*p,(x,-oo,oo))#calculul valorii mediei statistice a
    distributiei.
mu_x#afiseaza rezultatul
```

Rezultat: 0

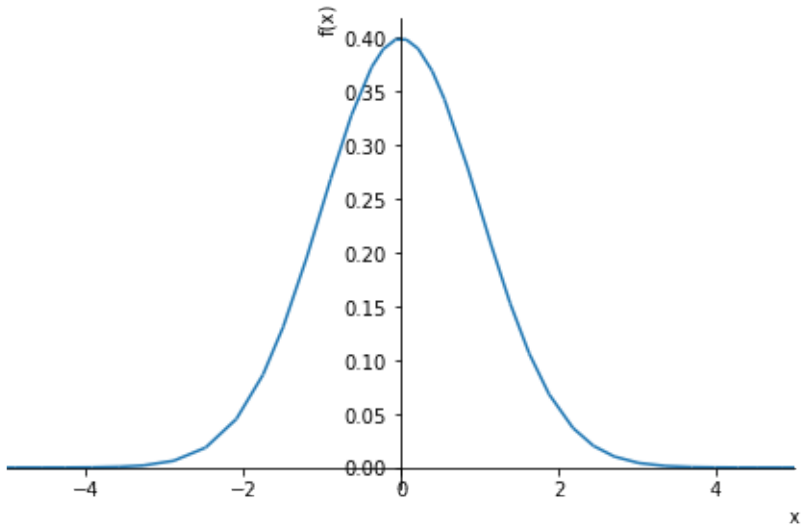
Se calculeaza valoarea varianței statistice a distribuției.

```
var_x=integrate((x-mu_x)**2*p,(x,-oo,oo))#calculul valorii
    variantei statistice a distributiei
var_x#afiseaza rezultatul
```

Rezultat: 1

Se reprezintă grafic funcția distribuție de probabilitate.

```
plot(p, xlim=(-5,5), show=True)#reprezentarea grafica a functiei p
    - densitatea de probabilitate
```



Graficul functiei $p(x)$

Se calculează probabilitatea ca valoarea variabilei aleatoare să fie cuprinsă în intervalul $(-3:3)$.

```
P1=integrate(p,(x,-oo,3)).evalf()#calculul probabilitatii ca
    valorile variabilei aleatoare sa fie mai mici dacat 3
P2=1-integrate(p,(x,-oo,-3)).evalf()#calculul probabilitatii ca
    valorile variabilei aleatoare sa fie mai mari dacat -3
P3=P1*P2#calculul probabilitatii ca valorile variabilei aleatoare
    sa fie in intervalul (-3; 3)
format(P3,'5.4')#afiseaza rezultatul
```

Rezultat: 0.9973

Problema 2.2

Etapele rezolvării.

(a) - Calculul valorii constantei c

(i) Condiția ca funcția $p(x)$ să reprezinte densitatea de probabilitate a unei variabile aleatoare este următoarea:

$$\int_{-\infty}^{\infty} p(x)dx = \int_{-\infty}^{\infty} \frac{c}{e^x + e^{-x}}dx = 1 \tag{2.1}$$

(ii) Pentru calculul integralei definite 2.1, mai întâi se calculeaza primitiva funcției cu substituția $y = e^x$.

$$\int \frac{c}{e^x + e^{-x}}dx = c \cdot \int \frac{dy}{y^2 + 1} = c \cdot \arctan y = c \cdot \arctan e^x + K \tag{2.2}$$

(iii) Valoarea integralei definite rezultă după cum urmează:

$$\int_{-\infty}^{\infty} p(x)dx = c \cdot \arctan e^x \Big|_{-\infty}^{+\infty} = c \cdot \frac{\pi}{2} \tag{2.3}$$

(iv) Valoarea constantei c se obține pe baza relațiilor 2.1 și 2.3

$$c \cdot \frac{\pi}{2} = 1$$

(b) - Calculul probabilității $P(X < 1, Y > 1)$

(i) Expresia funcției de repartiție de probabilitate, $P(x)$ se calculeaza prin inlocuirea valorii constantei c cu valoarea calculată la punctul precedent.

$$P(x) = \int_{-\infty}^{x} p(\xi)d\xi = \int_{-\infty}^{x} \frac{c}{e^\xi + e^{-\xi}}d\xi = \frac{2}{\pi} \cdot \arctan e^x.$$

(ii) Calculul probabilității condiționate din enunț se face cu expresia:

$$P(X < 1, Y > 1) = P(X < 1) \cdot P(Y > 1) = P(1) - [1 - P(1)].$$

Coduri - Aplicația 2.2.

Se apelează modulele necesare.

```
import numpy#modul pentru calcule numerice
import scipy#modul pentru calcule stiintifice
import sympy#modul pentru calcul simbolic
from sympy import*#importa toata colectia de functii ale modulului
from matplotlib import pyplot#importa colectia de functii pentru
    reperzentare grafica
sympy.init_printing()#comada pentru initializarea printarii
```

Se definesc variabilele simbolice ale aplicației.

```
c=symbols('c', positive=True)#definitia constantei c
x=symbols('x', real=True)#defintia variabilei x
y=symbols('y', positive=True)#defintia variabilei auxiliare y
```

```
F=Function('F')(y)#definitia functiei simbolice
F=sympy.integrate(1/(Pow(y,2)+1),(y,0,exp(x)))#expresia functiei
F#printeaza rezultatul
```

Rezultat: $\arctan(e^x)$

```
I=limit(F,x,oo)#calculul valorii la infinit a functiei repartitie
   de probabilitate
I#printeaza rezultatul
```

$I = \lim_{x\to\infty} F(x) = \frac{\pi}{2}$.

Pentru ca funcția $p(x)$ să reprezinte densitatea de probabilitate a unei variabile aleatoare, aceasta trebuie să verifice condițiile:

(i) $F(x)$ să fie pozitivă pe axa reală

(ii) $\int_{-\infty}^{+\infty} p(x)dx = 1$

Prima condiție implică $c > 0$.
A doua condiție este verificată dacă c este soluția ecuației asociate $c \cdot I - 1 = 0$.

```
c_num=sympy.solve(I*c-1,c)#rezolva ecuatia asociata
c_num[0]#printeaza rezultatul
```

Rezultat: $\frac{2}{\pi}$.

```
F1=expand(simplify(F*c_num[0]))
F1#printeaza rezultatul
```

Rezultat: $\frac{2\cdot\arctan(e^x)}{\pi}$

Calcul probabilității $P(X < 1)$

```
P_x1=N(expand(F1.subs(x,1)))#calculeaza valoarea numerica
format(P_x1, '5.3')#printeaza rezultatul
```

Rezultat: 0.776
Calcul probabilității $P(Y > 1)$

```
P_y1=1-N(expand(F1.subs(x,1)))#calculeaza valoarea numerica
format(P_y1,'5.3')#printeaza rezultatul
```

Rezultat: 0.224
Calculul probabilității condiționate $P(X < 1, Y > 1)$

```
P_xy1=P_x1*P_y1
format(P_xy1,'5.3')#printeaza rezultatul
```

Rezultat: 0.174

Problema 2.3

ETAPELE REZOLVĂRII.

(a) - Calculul valorilor mediei statistice și variației variabilei aleatoare X

1. Testarea primei condiții: $p(x) \geq 0 \forall x \in \mathbb{R}$ deoarece x^2 este pozitiv pe toată axa reală. Pentru testarea celei de-a doua condiții se calculează integrala $\int_{-\infty}^{+\infty} p(x)dx$.

2. Media statistică a variabilei aleatoare X se calculează prin mediere pe ansamblu. Se ține cont de expresia de definiție a funcției densitate de probabilitate din enunțul problemei.

$$\mu_x = E\{X\} = \int_{-\infty}^{+\infty} x \cdot p(x)dx = \int_{-1}^{+1} \frac{2}{\pi} \cdot \frac{x}{1+x^2}. \tag{2.4}$$

3. Varianța variabilei aleatoare X se calculează prin mediere pe ansamblu cu relația $\sigma_x^2 = \int_{-\infty}^{+\infty} (x-\mu_x)^2 \cdot p(x)dx$.

(b) - Calculul valorii mediei statistice a variabilei aleatoare Y

Media statistică asociate variabilei aleatoare Y se calculeaza cu relația **??** ținând cont de dependența dintre variabilele aleatoare X și Y dată în enunțul problemei.

$$\mu_y = E\{Y\} = E\{X^2\} = \int_{-\infty}^{+\infty} x^2 \cdot p(x)dx = \int_{-1}^{+1} \frac{2}{\pi} \cdot \frac{x^2}{1+x^2}. \tag{2.5}$$

OBSERVAȚIE: $\mu_y = \sigma_x^2$.

CODURI - APLICAȚIA 2.3.

Se apelează modulele necesare.

```
import numpy#modul pentru calcule numerice
import scipy#modul pentru calcule stiintifice
import sympy#modul pentru calcul simbolic
from sympy import*#importa toata colectia de functii ale modulului
from matplotlib import pyplot#importa colectia de functii pentru
    reprezentare grafica
sympy.init_printing()#comada pentru initializarea printarii
```

Definitiile variabilelor simbolice.

```
x=symbols('x', real=True)#defintia variabilei x
miu_x=symbols('miu_x', real=True)#defintia variabilei miu_x
var_x=symbols('var_x', real=True)#definitia variabilei var_x
miu_x2=symbols('miu_x2', real=True)#defintia variabilei miu_x2
```

Definitia functiei p(x).

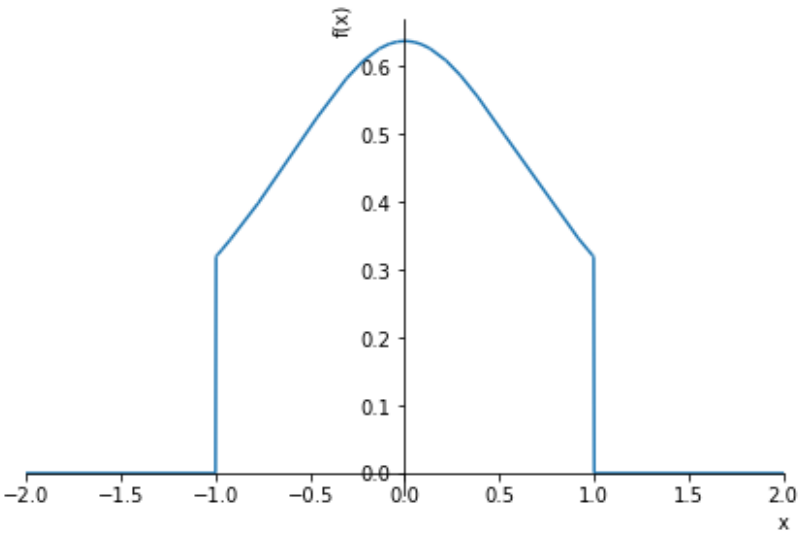
```
p=sympy.Piecewise((2/pi/(1+x**2), Abs(x)<1),(0, True))
p
```

Rezultat:

$$p(x) = \begin{cases} \frac{2}{\pi(x^2+1)} & \text{for } |x| < 1 \\ 0 & \text{otherwise} \end{cases}$$

Reprezentarea grafica a functiei p(x).

```
p1 = plot(p, xlim=(-2,2),show=True)
```



Graficul functiei $p(x)$.

Verificarea expresiei functiei p(x).

```
integrate(p,(x,-oo,oo))
```

Rezultat: 1

Calculul valorii mediei statistice.

```
miu_x=integrate(x*p,(x,-oo,oo))
miu_x
```

Rezultat: 0

Calculul valorii variantei statistice.

```
var_x=integrate(((x-miu_x)**2)*p,(x,-oo,oo))
var_x
```

Rezultat: $-1 + \frac{4}{\pi}$

Calculul valorii mediei statistice a variabilei aleatoare Y

```
miu_x2=integrate((x**2)*p,(x,-oo,oo))
miu_x2
```

Rezultat: $-1 + \frac{4}{\pi}$

3

Procese aleatoare și zgomote

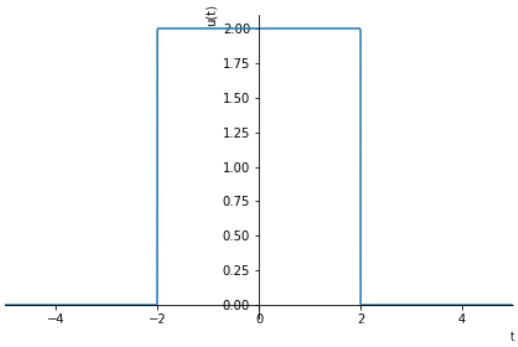
Enunțuri

Problema 3.1

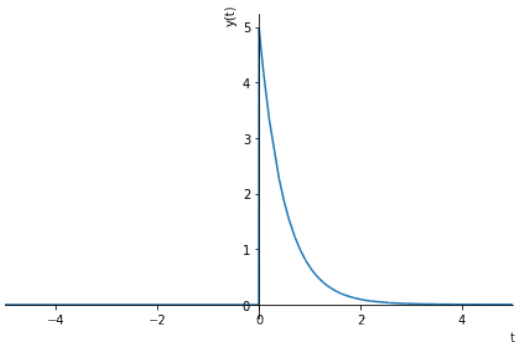
Să se calculeze expresia funcției de autocorelație a pulsului dreptunghiular cu amplitudinea U_M, reprezentat prin expresia care urmează:

$$u(t) = \begin{cases} U_M & \text{for } |t| < 2 \\ 0 & \text{otherwise} \end{cases} \quad \forall t \in \mathbb{R}.$$

Exemplu numeric: $U_M = 2$.



Reprezentarea grafică a pulsului din enunțul problemei 1.



Reprezentarea grafică a semnalului cauzal din enunțul problemei 2.

Problema 3.2

Să se calculeze expresia funcției de autocorelație pentru semnalul cauzal definit după cum urmează.

$$y(t) = \begin{cases} 0 & \text{for } t < 0 \\ U_M e^{-\lambda t} & \text{otherwise} \end{cases} \quad \lambda > 0.$$

Date numerice: $U_M = 5$, $\lambda = 2$.

Problema 3.3

O aproximare a funcției de autocorelație a unui proces aleator staționar de tip zgomot colorat este reprezentată prin expresia care urmează.

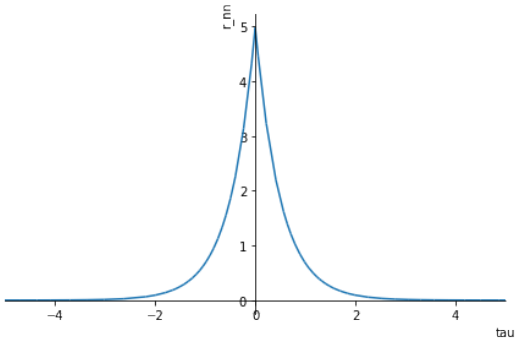
$$r_{uu}(\tau) = \sigma \cdot e^{-\lambda \cdot |\tau|} \quad \forall \tau \in \mathbb{R}; \lambda > 0.$$

Pe baza acestei aproximări, se cere să se estimeze:

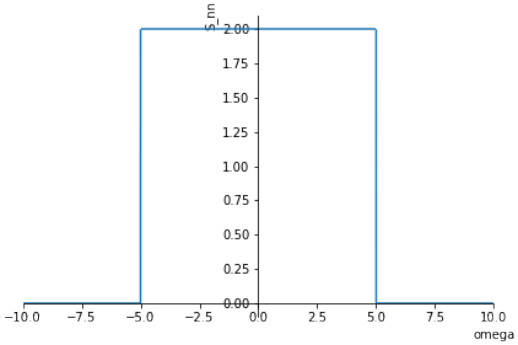
(a) Expresia funcției densitate spectrală de putere, $S_{uu}(\omega)$.

(b) Valoarea puterii procesului aleator.

Date numerice: varianța, $\sigma = 5$ și atenuarea, $\lambda = 2$.



Reprezentarea grafică a funcției de autocorelație a procesului aleator din enunțul problemei 3.3.



Reprezentarea grafică a funcției densitate spectrală de putere a procesului aleator din enunțul problemei 3.4.

Problema 3.4

Densitatea spectrală de putere a unui proces aleator staționar de tip zgomot alb cu bandă limitată este următoarea.

$$S_{nn}(\omega) = \begin{cases} N & \omega \in [-\Omega; +\Omega] \\ 0 & \omega \in (-\infty; -\Omega) \cup (+\Omega; +\infty) \end{cases}.$$

Se cere:

(a) Să se calculeze expresia funcției de autocorelație, $r_{nn}(\tau)$ a procesului aleator.

(b) Să se calculeze valoarea puterii procesului aleator.

(c) Să se calculeze valorile mediei statistice, μ_{nn} și varianței, σ^2_{nn} procesului aleator.

Date numerice: $\Omega = 5$ și $N = 2$.

Problema 3.5

Fie un semnal sinusoidal cu amplitudinea U_M, pulsația ω și faza inițială φ:

$$u(t) = U_M \cdot \sin(\omega_1 \cdot t + \varphi).$$

Se cer următoarele.

(a) Să se calculeze expresia funcției de autocorelație $r_{uu}(\tau)$.

(b) Să se calculeze expresia funcției densitate spectrală de putere $S_{uu}(\omega)$.

(c) Să se calculeze expresia puterii semnalului și să se arate că aceasta nu depinde de faza inițială a semnalului.

Date numerice: $U_M = 10$, $\omega_1 = 2$.

Problema 3.6

Să se calculeze expresia funcției de autocorelație pentru semnalul sinusoidal din problema precedentă dacă faza inițială a semnalului este un proces aleator cu repartiție uniformă caracterizată prin densitatea de probabilitate:

$$p(\varphi) = \begin{cases} \frac{1}{2\pi} & \text{dacă} \quad \varphi \in [-\pi; \pi) \\ 0 & \text{dacă} \quad \varphi \in (-\infty; -\pi) \cup [\pi; +\infty) \end{cases}.$$

Date numerice: $U_M = 10$, $\omega_1 = 2$.

Problema 3.7

Într-un experiment de identificare a fost analizat un proces aleator staționar cu media statistică zero. S-a constatat că funcția de autocorelație a acestuia poate fi aproximată prin expresia următoare.

$$R_{yy}(\tau) = \sigma^2 \cdot e^{-|\tau|} \cdot cos(\tau) \;\; \forall \tau \in \mathbb{R};.$$

Se cere:

(a) să se calculeze expresia funcției densitate spectrală de putere, $S_{yy}(\omega)$ a procesului aleator,

(b) să se demonstreze că puterea medie a procesului aleator este numeric egală cu parametrul σ^2,

(c) să se calculeze valoarea pulsației ω_M care corespunde valorii maxime a funcției $S_{yy}(\omega)$

Date numerice: $\sigma^2 = 40$.

Răspunsuri

Problema 3.1:

$-4\max(-2,-\tau-2)+4\max(-2,-\tau-2,\min(2,2-\tau))$.
sau:

$$R_{yy}(\tau)=\begin{cases}0 & \text{dacă } \tau\in(-\infty;-4)\\ 4\tau+16 & \text{dacă } \tau\in[-4;0)\\ -4\tau+16 & \text{dacă } \tau\in[0;4)\\ 0 & \text{dacă } \tau\in[4;\infty)\end{cases}.$$

Problema 3.2:

$R_{yy}(\tau)=\frac{25e^{-2\tau}e^{-4\max(0,-\tau)}}{4}$ și $R_{yy}(0)=6.25$.

Problema 3.3:

(a) $S_{uu}(\omega)=\frac{20}{\omega^2+4}$.

(b) $P=5.0$

Problema 3.4:

(a) $r_{nn}(\tau)=\begin{cases}\frac{2.0\sin(5\tau)}{\pi\tau} & \text{for } \tau\neq 0\\ \frac{10.0}{\pi} & \text{otherwise}\end{cases}$; **(b)** $P_{nn}=\frac{10.0}{\pi}$; **(c)** $\mu_{nn}=0$, $\sigma_{nn}^2=\frac{10.0}{\pi}$.

Problema 3.5:

(a) $R_{uu}(\tau)=50\cdot\cos(2\tau)$; **(b)** $S_{uu}(\omega)=50\pi\left[\delta(\omega-2)+\delta(\omega+2)\right]$; **(c)** $P_{uu}=50$.

Problema 3.6:

$R_{uu}(\tau)=50\cdot\cos(2\tau)$.

Problema 3.7:

(a) $S_{yy}(\omega)=\frac{2\sigma^2\left(\omega^2+2\right)}{\omega^4+4}$; **(b)** $P_{yy}=\sigma^2$; **(c)** $\omega_M=0.91$.

Rezolvări

Rezolvarea problemei 3.1

Etapele rezolvării.
Pulsul $u(t)$ este de tip necauzal. Funcția de autocorelație a pulsului $u(t)$ se calculează pe tot $\mathbb{R}$ cu formula $r_{uu}(\tau) = \int_{-\infty}^{\infty} u(t) \cdot u(t+\tau)dt$.

Coduri - Aplicația 3.1.
Se apelează modulele necesare.

```
import sympy
import numpy
import scipy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Definițiile variabilelor și funcțiilor.

```
tau=symbols('tau', real=True)#decalajul in timp
U_M=symbols('U_M',positive=True)#variabila amplitudine a semnalului
    dreptunghiular
t=symbols('t',real=True)#variabila t - argumentul functiei-semnal
    dreptunghiular
u=sympy.Function('u')(t)#functia-semnal dreptunghiular
u_tau=sympy.Function('u_tau')(t)#definitia functiei semnal-
    dreptunghiular decalata cu tau
r_xx=sympy.Function('r_xx')(tau)#definitia functiei de
    autocorelatie
```

Expresia funcției-semnal.

```
u=Piecewise((U_M,abs(t)<2),(0,True))#expresia functiei-semnal
u#afiseaza rezultatul
```

Rezultat:

$$\begin{cases} U_M & \text{for } |t| < 2 \\ 0 & \text{otherwise} \end{cases}$$

Expresia funcției-semnal translatată cu τ.

```
u_tau=Piecewise((U_M,abs(tau+t)<2),(0,True))#expresia functiei-
    semnal translatate cu tau
u_tau#afiseaza rezultatul
```

Rezultat:

$$\begin{cases} U_M & \text{for } |t+\tau| < 2 \\ 0 & \text{otherwise} \end{cases}$$

Calculul expresiei funcției de autocorelație.

```
r_uu=sympy.integrate(u*u_tau,(t,-oo,oo))#calculul functiei de
    autocorelatie
r_uu#afiseaza rezultatul
```

Rezultat:

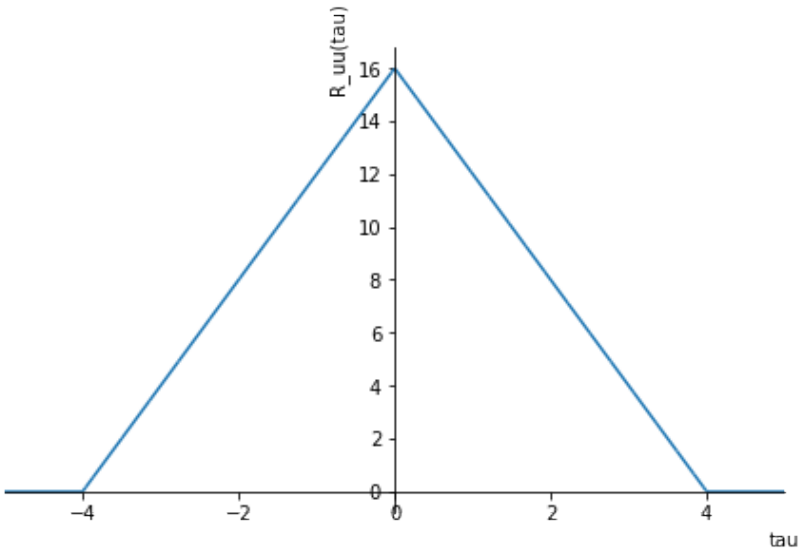
$$-U_M^2 \max(-2, -\tau - 2) + U_M^2 \max(-2, -\tau - 2, \min(2, 2 - \tau))$$

Reprezentarea grafică.

```
dU_M=2#valoarea numerica a variabilei U_M
dU=u.subs(U_M,dU_M)
plot_parametric((t,dU),xlabel='t',ylabel='u(t)',xlim=(-5,5),show=
    True)
```

```
dR_uu=r_uu.subs(U_M,dU_M)
plot(dR_uu,xlabel='tau', ylabel='R_uu(tau)',xlim=(-5,5),show=True)
```

Rezultat:



Graficul funcției de autocorelație a pulsului dreptunghiular cu amplitudinea $U_M = 5$.

Rezolvarea problemei 3.2

ETAPELE REZOLVĂRII.
Se procedează analog cu problema percedendă.
În acest caz, funcția-semnal $y(t)$ este de tip cauzal deci, la calculul integralei se ține cont că funcția semnal este diferită de zero doar în intervalul $t \in (0; \infty)$.

CODURI - APLICAȚIA 3.2.
Se apelează modulele necesare.

```
import sympy
import numpy
import scipy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Se definesc variabilele și funcțiile simbolice.

```
t=symbols('t',real=True)#defintia variabilei t - argumentul
    functiei
tau=symbols('tau', real=True)#defintia decalajului in timp
lam=symbols('lambda', positive=True)#defintia parametrului lambda
Y_M=symbols('Y_M',positive=True)#definitia variabilei amplitudine a
    semnalului sinusoidal
y=Function('y')(t)#definitia functiei semnal sinusoidal
y_tau=Function('y_tau')(t)#definitia functiei semnal sinusoidal
    decalata cu tau
r_yy=Function('r_yy')(tau)#definitia functiei de autocorelatie
```

Se definesc valorile numerice ale parametrilor, propuse în enunțul problemei.

```
dY_M=5#valoarea numerica a parametrului U_M
dLam=2#valoarea numerica a parametrului lambda
```

Se definește expresia funcției-semnal.

```
y=Piecewise((0,t<0),(Y_M*sympy.exp(-lam*t),t>=0))#expresia functiei
    -semnal; definita pe intervale
y#afiseaza rezultatul
```

Rezultat:

$$\begin{cases} 0 & \text{for } t < 0 \\ Y_M e^{-\lambda t} & \text{otherwise} \end{cases}$$

Se definește expresia funcției-semnal translatată cu τ.

```
y_tau=Piecewise((0,t+tau<0),(Y_M*sympy.exp(-lam*(t+tau)),t+tau>=0))
    #expresia functiei-semnal translatate cu tau; definita pe
    intervale
y_tau#afiseaza rezultatul
```

Rezultat:

$$\begin{cases} 0 & \text{for } t + \tau < 0 \\ Y_M e^{-\lambda(t+\tau)} & \text{otherwise} \end{cases}$$

Se calculează expresia funcției de autocorelație, $r_{yy}(\tau)$.

```
r_yy=sympy.integrate(y*y_tau,(t,-oo,oo))#calculul functiei de
    autocorelatie
r_yy#afiseaza rezultatul
```

Rezultat:

$$\frac{Y_M^2 e^{-\lambda\tau} e^{-2\lambda \max(0,-\tau)}}{2\lambda}$$

Se substituie valorile numerice în expresia funcției de autocorelație și se reprezintă graficul acesteia.

```
dR_yy=r_yy.subs(Y_M,dY_M).subs(lam,dLam)
plot(dR_yy,xlabel='tau', ylabel='R_yy(tau)',xlim=(-5,5),show=True)
```

Rezultat:

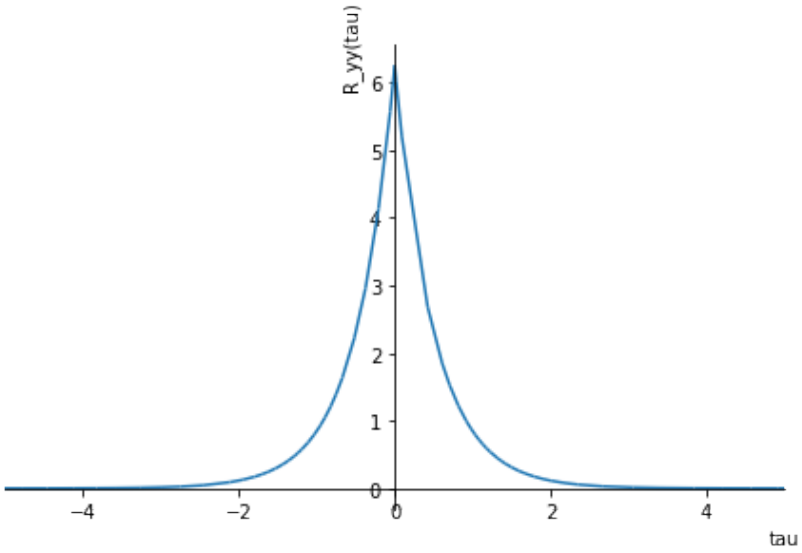


Figura 3.1: Graficul funcției de autocorelație a semnalului cauzal din enunțul problemei 2 pentru $U_M = 5$ și $\lambda = 2$. Valoarea maximă pe grafic corespunde la $r_{yy}(0) = 6.25$.

Rezolvarea problemei 3.3

ETAPELE REZOLVĂRII.
(a) -
CODURI - APLICAȚIA 3.3.
Se apelează modulele necesare.

```
import sympy
import numpy
import scipy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Se definesc variabilele și funcțiile simbolice

```
tau=symbols('tau',real=True)#defintia variabilei tau - decalajul in
    timp
omega=symbols('omega', positive=True)#defintia variabilei omega -
    pulsatia
f=symbols('f',positive=True)#definitia variabilei frecventa
sigma=symbols('sigma', positive=True)#defintia parametrului sigma -
    varianta
lam=symbols('lambda', positive=True)#definitia parametrului lambda
   - atenuarea
r_nn=Function('r_nn')(tau)#definitia functiei de autocorelatie
S_nn=Function('S_nn')(omega)#definitia functiei densitate spectrala
    de putere
```

```
dSigma=5#valoarea numerica a parametrului sigma
dLam=2#valoarea numerica a parametrului lambda
```

Se definește expresia funcției de autocorelație.

```
r_nn=Piecewise((sigma*exp(lam*tau),tau<0),(sigma*exp(-lam*tau),tau
   >=0))#expresia de autocorelatie; definita pe intervale
r_nn#afiseaza rezultatul
```

Rezultat:

$$\begin{cases} \sigma e^{\lambda\tau} & \text{for } \tau < 0 \\ \sigma e^{-\lambda\tau} & \text{otherwise} \end{cases}$$

Se definește expresia funcției de autocorelație pe intervale.

```
r1_nn=sigma*exp(lam*tau)#expresia functiei de autocorelatie pe
    intervalul (-oo;0)
r2_nn=sigma*exp(-lam*tau)#expresia functiei de autocorelatie pe
    intervalul [0;+oo)
r1_nn,r2_nn
```

Rezultat: $r1_{nn}(\tau) = \sigma \cdot e^{\lambda\tau}$; $r2_{nn}(\tau) = \sigma \cdot e^{-\lambda\tau}$.

Se calculează expresia funcției densitate spectrală de putere prin integrare pe intervale.

```
S_nn=(integrate(r1_nn*exp(-I*omega*tau),(tau,-oo,0))+integrate(
    r2_nn*exp(-I*omega*tau),(tau,0,oo)))#expresia functiei densitate
     spectrala de putere#calculeaza densitatea spectrala de putere
simplify(S_nn)#afiseaza rezultatul
```

Rezultat:

$$S_{nn}(\omega) = \frac{2\lambda\sigma}{\lambda^2 + \omega^2}$$

Se calculează expresia puterii procesului aleator.

```
P=1/2/pi*integrate(S_nn,(omega,-oo,oo))#puterea procesului aleator
    calculata pe baza densitatii spectrale de putere
P#afiseaza rezultatul
```

Rezultat: $P = \sigma$.

Se calculează expresia densității spectrale de putere cu valori numerice și se reprezintă grafic.

```
dS_nn=S_nn.subs(sigma,dSigma).subs(lam,dLam)#pentru reprezentarea
    grafica, se rescrie expresia cu valorile date in enunt
plot(dS_nn,xlabel='omega',ylabel='S_nn',xlim=(-10,10),show=True)
```

```
simplify(dS_nn)#expresia functiei densitate spectrala de putere cu
    valori numerice
```

Rezultat: $S_{nn}(\omega) = \frac{20}{\omega^2+4}$.

Pe baza expresiei funcției densitate spectrală de putere, se calculează expresia puterii procesului aleator cu valori numerice.

```
P=1/2/pi*integrate(dS_nn,(omega,-oo,oo))#puterea procesului aleator
     calculata pe baza densitatii spectrale de putere
P#afiseaza rezultatul
```

Rezultat: $P = 5$.

OBSERVAȚIE: Puterea procesului aleator depinde de varianța procesului, σ dar nu depinde de atenuarea λ.

Rezolvarea problemei 3.4

Etapele rezolvării.

(a) - Calculul expresiei funcției de autocorelație.

Funcția de autocorelație este numeric egală cu transformata Fourier inversă a funcției densitate spectrală de putere; prin urmare:

$$r_{nn}(\tau) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} S_{nn}(\omega)\cdot e^{j\omega\tau}d\omega = \frac{1}{2\pi}\int_{-\Omega}^{+\Omega} N\cdot e^{j\omega\tau}d\omega = \frac{N\sin(\Omega\tau)}{\pi\tau}.$$

.

Observație: valoarea funcției de autocorelație în punctul $\tau = 0$.

$$r_{nn}(0) = \frac{N\Omega}{\pi}\cdot\lim_{\Omega\tau\to 0}\frac{\sin(\Omega\tau)}{\Omega\tau} = \frac{N\Omega}{\pi}.$$

(b) - Calculul valorii puterii procesului aleator.

Valoarea puterii procesului aleator se calculează pe baza expresiei funcției densitate spectrală de putere cu formula următoare:

$$P = \frac{1}{2\pi}\cdot\int_{-\infty}^{+\infty} S_{nn}(\omega)d\omega = \frac{1}{2\pi}\cdot\int_{-\Omega}^{+\Omega} N d\omega = \frac{N\Omega}{\pi}.$$

Observație: $P = r_{nn}(0)$.

(c) - Calculul valorilor mediei statistice și a variației procesului aleator

Valoarea mediei statistice se calculează pe baza expresiei funcției de autocorelație cu formula:

$$\mu_{nn}^2 = \lim_{\tau\to\infty} r_{nn}(\tau) = \lim_{\tau\to\infty}\frac{1.0N\sin(\Omega\tau)}{\pi\tau} = 0.$$

Valoarea variației procesului aleator se obține pe baza valorii funcției de autocorelație pentru $\tau = 0$ și a expresiei:

$$\sigma_{nn}^2 + \mu_{nn}^2 = r_{nn}(0) = \frac{1.0N\Omega}{\pi},$$

în care $\mu_{nn} = 0$ conform rezultatului de mai sus.

Coduri - Aplicația 3.4.

Se apelează modulele necesare.

```
import sympy
import numpy
import scipy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Se definesc variabilele și funcțiile simbolice

```
tau=symbols('tau',real=True)#defintia variabilei tau - decalajul in
    timp
omega=symbols('omega', real=True)#defintia variabilei omega -
    pulsatia
f=symbols('f',positive=True)#definitia variabilei frecventa
```

```
N=symbols('N', positive=True)#defintia parametrului N
Omega=symbols('Omega',positive=True)#definitia parametrului Omega
r_nn=Function('r_nn')(tau)#definitia functiei de autocorelatie
S_nn=Function('S_nn')(omega)#definitia functiei densitate spectrala
    de putere
mu_n=symbols('mu_n', positive=True)#defintia mediei statistice a
    procesului aleator
var_n=symbols('var_n',positive=True)#defintia variantei statistice
    a procesului aleator
```

```
dN=2#valoarea numerica a parametrului N
dOmega=5#valoarea numerica a parametrului Omega
```

Se definește expresia funcției $S_{nn}(\omega)$

```
S_nn=Piecewise((N,abs(omega)<Omega),(0,True))#expresia functiei
    densitate spectrala de putere; definita pe intervale
S_nn#afiseaza rezultatul
```

Rezultat: $\begin{cases} N & \text{for } \Omega > |\omega| \\ 0 & \text{otherwise} \end{cases}$.

Se calculează expresia funcției $r_{nn}(\tau)$

```
r_nn=simplify(1/2/pi*integrate(N*exp(I*omega*tau),(omega,-Omega,
    Omega)))#calculul functiei de autocorelatie
r_nn#afisare rezultat
```

Rezultat: $\begin{cases} \frac{1.0N\sin(\Omega\tau)}{\pi\tau} & \text{for } \tau \neq 0 \\ \frac{1.0N\Omega}{\pi} & \text{otherwise} \end{cases}$.

Calculul expresiei puterii procesului aleator.

```
P=simplify(1/2/pi*integrate(N,(omega,-Omega,Omega)))#expresia
    puterii procesului aleator
P#afiseaza rezultatul ; acelasi rezultat si cu formula P = r_nn(0)
```

Rezultat: $\frac{1.0N\Omega}{\pi}$.

Calculele cu datele numerice din enunțul problemei.

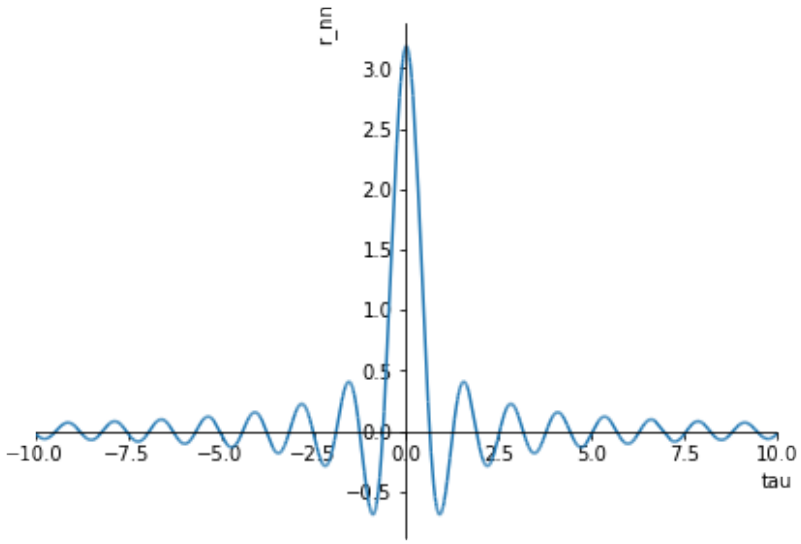
```
dS_nn=S_nn.subs(Omega,dOmega).subs(N,dN)
dR_nn=r_nn.subs(Omega,dOmega).subs(N,dN)
dP=P.subs(Omega,dOmega).subs(N,dN)
dS_nn,dR_nn,dP
```

Rezultat: $\begin{cases} 2 & \text{for } |\omega| < 5 \\ 0 & \text{otherwise} \end{cases}$; $\begin{cases} \frac{2.0\sin(5\tau)}{\pi\tau} & \text{for } \tau \neq 0 \\ \frac{10.0}{\pi} & \text{otherwise} \end{cases}$; $\frac{10.0}{\pi}$.

Reprezentarea grafică a funcției de autocorelație.

```
plot(dR_nn,xlabel='tau',ylabel='r_nn',xlim=(-10,10),show=True)
```

Rezultat:



Graficul funcției de autocorelație a procesului aleator de tip zgomot alb cu bandă limitată. $N = 2$,$\Omega = 5$.

Rezolvarea problemei 3.5

Etapele rezolvării.

(a) - Calculul expresiei funcției de autocorelație.

Se utilizează relația **??** în care se ține cont că funcția semnal este periodică cu perioada $T_1 = \frac{2\pi}{\omega_1}$:

$$r_{uu}(\tau) = \frac{1}{2T_1} \cdot \int_{-T_1}^{+T_1} u(t) \cdot u(t+\tau)dt = \frac{1}{2T_1} \cdot \int_{-T_1}^{+T_1} U_M^2 \sin(\omega_1 t + \varphi) \cdot \sin[\omega_1(t+\tau) + \varphi]dt$$

Rezultă: $r_{uu}(\tau) = \frac{U_M^2}{2}\cos(\omega_1\tau)$.

(b) - Calculul expresiei funcției densitate spectrală de putere.

Funcția densitate spectrală de putere este egală cu transformata Fourier a funcției de autocorelație. Prin urmare:

$$\begin{aligned}
S_{uu}(\omega) &= \int_{-\infty}^{+\infty} r_{uu}(\tau)e^{-j\cdot\omega\cdot\tau}d\tau \\
&= \int_{-\infty}^{+\infty} \frac{U_M^2}{2}\cos(\omega_1\tau)e^{-j\cdot\omega\cdot\tau}d\tau \\
&= \frac{U_M^2}{2}\int_{-\infty}^{+\infty} \frac{e^{j\omega_1\tau} + e^{-j\omega_1\tau}}{2}e^{-j\omega\tau}d\tau \\
&= \frac{U_M^2}{4}\left[\int_{-\infty}^{+\infty} e^{j\omega_1\tau}e^{-j\omega\tau}d\tau + \int_{-\infty}^{+\infty} e^{-j\omega_1\tau}e^{-j\omega\tau}d\tau\right] \\
&= \frac{U_M^2}{4}\left[\int_{-\infty}^{+\infty} e^{-j(-\omega_1+\omega)\tau}d\tau + \int_{-\infty}^{+\infty} e^{-j(\omega_1+\omega)\tau}d\tau\right].
\end{aligned}$$

Se ține cont că transformata Fourier a unei constante C este:

$$\mathcal{F}(C) = \int_{-\infty}^{+\infty} Ce^{-j\omega\tau}d\tau = 2\pi \cdot C \cdot \delta(\omega)$$

în care $\delta(\omega)$ este impulsul Dirac și, rezultă:

$$S_{uu}(\omega) = \frac{\pi \cdot U_M^2}{2}\left[\delta(\omega - \omega_1) + \delta(\omega + \omega_1)\right].$$

(c) - Calculul expresiei puterii semnalului sinusoidal.

Puterea semnalului sinusoidal se calculează prin integrarea funcției densitate spectrală de putere pe tot domeniul frecvențelor astfel:

$$P_{uu} = \frac{1}{2\pi}\int_{-\infty}^{+\infty} S_{uu}(\omega)d\omega = \frac{U_M^2}{4} \cdot \int_{-\infty}^{+\infty} \left[\delta(\omega - \omega_1) + \delta(\omega + \omega_1)\right]d\omega.$$

Coduri - Aplicația 3.5.

Se definesc variabilele și funcțiile simbolice.

```
import sympy
import numpy
import scipy
```

```
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Definițiile variabilelor simbolice și funcțiilor.

```
t=symbols('t',real=True)#variabila timp
tau=symbols('tau', real=True)#variabila tau - decalajul in timp
omega=symbols('omega', real=True)#variabila pulsatie
U_M=symbols('U_M',positive=True)#amplitudinea semnalului sinusoidal
omega_1=symbols('omega_1',positive=True)#pulsatia semnalului
    sinusoidal
phi=symbols('phi', real=True)#faza initiala a semnalului sinusoidal
u=sympy.Function('u')(t)#functia semnal sinusoidal
u_aux=sympy.Function('u_aux')(t)#functia semnal sinusoidal
u_tau=sympy.Function('u_tau')(t)#functia semnal sinusoidal decalata
    cu tau
u_tau_aux=sympy.Function('u_tau_aux')(t)#functia semnal sinusoidal
    decalata cu tau
T_1=symbols('T_1', integer=True)#perioada semnalului sinusoidal
r_uu=sympy.Function('r_uu')(tau)#functia de autocorelatie
S_uu=sympy.Function('S_uu')(omega)#functia densitate spectrala de
    putere
```

Valorile numerice ale parametrilor.

```
dU_M=10#valoarea numerica a amplitudinii semnalului sinusoidal
dOmega_1=2#valoarea numerica a pulsatiei semnalului sinusoidal
```

Expresiile funcțiilor semnal sinusoidal si semnal sinusoidal decalat.

```
u=U_M*sin(omega_1*t+phi)#expresia functiei semnal sinusoidal
u_aux=expand_trig(expand(u))#expresia desfasurata a functiei semnal
    sinusoidal
u,u_aux#afiseaza rezultatul
```

Rezultat:

$U_M \sin(\omega_1 t + \phi)$

$U_M (\sin(\phi)\cos(\omega_1 t) + \sin(\omega_1 t)\cos(\phi))$

```
u_tau=U_M*sin(omega_1*(t+tau)+phi)#expresia functiei semnal
    sinusoidal decalata cu tau
u_tau_aux=expand_trig(expand(u_tau))#expresia desfasurata a
    functiei semnal sinusoidal decalata cu tau
u_tau,u_tau_aux#afiseaza rezultatul
```

Rezultat: $U_M \sin(\omega_1 (t+\tau) + \phi)$.

Calculul expresiei funcției de autocorelație.

```
T_1=2*pi/omega_1#perioada semnalului sinusoidal
r_uu=1/2/T_1*sympy.integrate(expand(u_aux*u_tau_aux),(t,-T_1,T_1))#
    calculul expresiei functiei de autocorelatie a semnalului
    sinusoidal
```

```
r_uu_aux=simplify(r_uu)#variabila auxiliara
r_uu_aux#afiseaza rezultatul
```

Rezultat: $0.5U_M^2 \cos(\omega_1 \tau)$.

Expresia funcției densitate spectrală de putere calculată analitic.

```
S_uu=sympy.pi/2*U_M**2*(sympy.DiracDelta(omega-omega_1)+sympy.
    DiracDelta(omega+omega_1))
S_uu#afiseaza rezultatul
```

Rezultat: $\frac{\pi U_M^2(\delta(\omega-\omega_1)+\delta(\omega+\omega_1))}{2}$.

Calculul expresiei puterii semnalului sinusoidal.

```
P_uu=1/2/pi*integrate(S_uu,(omega,-oo,oo))#expresia puterii
    semnalului sinusoidal
P_uu#afiseaza rezultatul
```

Rezultat: $0.5U_M^2$.

Observație: Rezultatul de mai sus demonstrează că puterea semnalului depinde doar de valoarea parametrului U_M.

Calcule cu valorile numerice din enunțul problemei.

```
dR_uu=simplify(r_uu.subs(U_M,dU_M).subs(omega_1,dOmega_1))#expresia
    functiei de autocorelatie cu valori numerice
dR_uu#afiseaza rezultatul
```

Rezultat: $50.0 \cos(2\tau)$.

```
dS_uu=S_uu.subs(U_M,dU_M).subs(omega_1,dOmega_1)#expresia functiei
    densitate spectrala de putere cu valori numerice
dS_uu#afiseaza rezultatul
```

Rezultat: $50\pi(\delta(\omega-2)+\delta(\omega+2))$.

```
dP_uu=P_uu.subs(U_M,dU_M).subs(omega_1,dOmega_1)#expresia puterii
    semnalului sinusoidal cu valori numerice
dP_uu#afiseaza rezultatul
```

Rezultat: 50.

Rezolvarea problemei 3.6

Etapele rezolvării.

Deoarece faza inițială a semnalului sinusoidal este un proces aleator, semnalul sinusoidal nu mai este determinist ci tot un proces aleator. Funcția de autocorelație se calculează prin mediere pe ansamblul valorilor procesului aleator după cum urmează.

$$\begin{aligned} r_{uu}(\tau) &= \int_{-\infty}^{\infty} u(t,\varphi)\cdot u(t+\tau,\varphi)\cdot p(\varphi)d\varphi \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} U_M^2 \sin(\omega_1 t+\varphi) sin[\omega_1(t+\tau)+\varphi]d\varphi \end{aligned}$$

Coduri - Aplicația 3.6.
Se definesc variabilele și funcțiile simbolice.

```
import sympy
import numpy
import scipy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Definițiile variabilelor și funcțiilor simbolice.

```
t=symbols('t',real=True)#variabila timp
tau=symbols('tau', real=True)#variabila tau - decalajul in timp
omega=symbols('omega', positive=True)#variabila pulsatie
omega_1=symbols('omega_1', positive=True)#pulsatia semnalului
    sinusoidal
U_M=symbols('U_M',positive=True)#amplitudinea semnalului sinusoidal
phi=symbols('phi', real=True)#faza initiala a semnalului sinusoidal
```

```
u=sympy.Function('u')(t)#functia semnal sinusoidal
u_aux=sympy.Function('u_aux')(t)#functia semnal sinusoidal
u_tau=sympy.Function('u_tau')(t)#functia semnal sinusoidal decalat
    cu tau
u_tau_aux=sympy.Function('u_tau_aux')(t)#functia semnal sinusoidal
    decalat cu tau
T_1=symbols('T_1', positive=True)#perioada semnalului sinusoidal
p=sympy.Function('p')(phi)#functia densitate spectrala de putere a
    fazei initiale
r_uu=sympy.Function('r_uu')(omega)#functia de autocorelatie a
    semnalului sinusoidal
```

Valorile numerice ale parametrilor.

```
dU_M=10#valoarea numerica a amplitudinii semnalului sinusoidal
dOmega_1=2#valoarea numerica a pulsatiei semnalului sinusoidal
```

Expresiile funcțiilor.

```
u=U_M*sin(omega_1*t+phi)#expresia functiei semnal sinusoidal
p=Piecewise((1/(2*pi), abs(phi)<pi),(0, True))#expresia functiei
    densitate spectrala de putere a fazei initiale
u_aux=expand_trig(expand(u))#expresia desfasurata a functiei semnal
     sinusoidal
p,u,u_aux#afiseaza rezultatul
```

Rezultat:

$$p(\varphi) = \begin{cases} \frac{1}{2\pi} & \text{for } |\phi| < \pi \\ 0 & \text{otherwise} \end{cases},$$

$u(t) = U_M \sin(\omega_1 t + \phi)$.

```
u_tau=U_M*sin(omega_1*(t+tau)+phi)#expresia functiei semnal
    sinusoidal decalata cu tau
u_tau_aux=expand_trig(expand(u_tau))#expresia desfasurata a
    functiei semnal sinusoidal decalata cu tau
u_tau,u_tau_aux#afiseaza rezultatul
```

Rezultat: $u(t+\tau) = U_M \sin(\omega_1 (t+\tau) + \phi)$.

Calculul expresiei funcției de autocorelație.

```
r_uu=sympy.integrate(u_aux*u_tau_aux*p,(phi,-oo,oo))
simplify(r_uu)
```

Rezultat: $\frac{U_M^2 \cos(\omega_1 \tau)}{2}$

Calcule cu valorile numerice din enunțul problemei.

```
dR_uu=r_uu.subs(U_M,dU_M).subs(omega_1,dOmega_1)
simplify(dR_uu)#afiseaza rezultatul
```

Rezultat: $50\cos(2\tau)$.

OBSERVAȚIE. Expresia funcției de autocorelație este aceeași cu expresia funcției de autocorelație a semnalului sinusoidal determinist din problema precedentă. Explicația fiind aceea că funcția de autocorelație a semnalului sinusoidal nu depinde de faza inițială a acestuia.

Rezolvarea problemei 3.7

Etapele rezolvării.

(a) - Calculul expresiei funcției densitate spectrală de putere.
Densitatea spectrală de putere este numeric egală cu transformata Fourier a funcției de autocorelație, prin urmare vom folosi relațiile:

$$\begin{aligned} S_{yy}(\omega) &= \mathcal{F}\left[R_{uu}(\tau)\right] = \int_{-\infty}^{\infty} R_{uu}(\tau)\cdot e^{-j\omega\tau}d\tau \\ &= \int_{-\infty}^{0} \sigma^2 e^{\tau}\cos\tau\cdot e^{-j\omega\tau}d\tau + \int_{0}^{\infty} \sigma^2 e^{-\tau}\cos\tau\cdot e^{-j\omega\tau}d\tau. \end{aligned}$$

(b) - Expresia puterii medii a procesului aleator.

Calculul se poate face în două moduri: (a) cu expresia funcției de autocorelație și proprietatea $P_{yy} = R_{yy}(0)$ sau (b) cu expresia funcției densitate spectrală de putere, prin integrare pe întreg domeniul pulsațiilor și relația:

$$P_{yy} = \frac{1}{2\pi}\cdot\int_{-\infty}^{\infty} S_{yy}(\omega)d\omega.$$

(c) - Calculul valorii pulsației care corespunde valorii maxime a funcției densitate spectrală de putere.

Se calculează expresia derivatei funcției $S_{yy}(\omega)$ în raport cu variabila și se rezolvă ecuația:

$$\frac{dS_{yy}(\omega)}{d\omega} = 0.$$

Cerința problemei se verifică pentru valorile pozitive ale soluțiilor ecuației de mai sus.

Coduri - Aplicația 3.7.
Se definesc variabilele și funcțiile simbolice.

```
import numpy
import sympy
from sympy import*
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor

```
sigma2=sympy.symbols('sigma^2',positive=True)#constanta sigma
tau=sympy.symbols('tau',real=True)#variabila tau - decalajul in
    timp
omega=sympy.symbols('omega', positive=True)#variabila omega -
    pulsatia
R_yy=sympy.Function('R_yy')(tau)#functia de autocorelatie a
    procesului aleator la portul de iesire al filtrului
S_yy=sympy.Function('S_yy')(omega)#functia densitate spectrala de
    putere a procesului aleator la portul de iesire
```

Expresia funcției de autocorelație a procesului aleator.

```
R_yy=sympy.Piecewise((sigma2*sympy.exp(tau)*sympy.cos(tau),tau<0)
                     ,(sigma2*sympy.exp(-tau)*sympy.cos(tau),tau
                         >=0))
R_yy
```

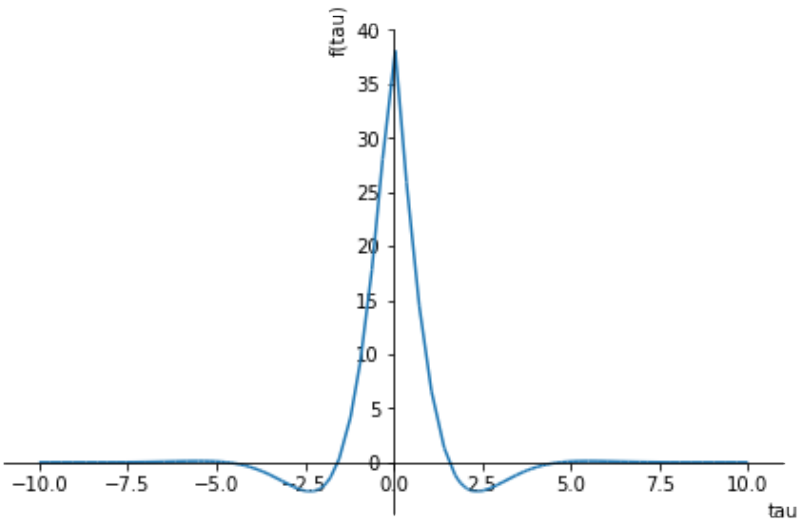
Rezultat: $R_{yy}(\tau) = \begin{cases} \sigma^2 e^{\tau}\cos(\tau) & \text{for } \tau < 0 \\ \sigma^2 e^{-\tau}\cos(\tau) & \text{otherwise} \end{cases}$.

```
dSigma2=40#valoarea numerica a parametrului sigma
dRyy=R_yy.subs(sigma2,dSigma2)
dRyy
```

Rezultat: $R_{yy}(\tau) = \begin{cases} 40e^{\tau}\cos(\tau) & \text{for } \tau < 0 \\ 40e^{-\tau}\cos(\tau) & \text{otherwise} \end{cases}$

Reprezentare grafică.

```
sympy.plot(dRyy,(tau,-10,10))
```



Graficul funcției de autocorelație a procesului aleator din enunțul problemei 3.7; $\sigma^2 = 40$.

Expresia funcției de autocorelație în intervalul $\tau < 0$.

```
dRyy1=((sigma2*sympy.exp(tau)*sympy.cos(tau)).rewrite(exp))
dRyy1exp=sympy.simplify(dRyy1).expand()
dRyy1exp
```

Rezultat: $\frac{\sigma^2 e^{\tau} e^{i\tau}}{2} + \frac{\sigma^2 e^{\tau} e^{-i\tau}}{2}$.

Expresia funcției de autocorelație în intervalul $\tau > 0$.

```
dRyy2=((sigma2*sympy.exp(-tau)*sympy.cos(tau)).rewrite(exp))
dRyy2exp=sympy.simplify(dRyy2).expand()
dRyy2exp
```

Rezultat: $\frac{\sigma^2 e^{-\tau} e^{i\tau}}{2} + \frac{\sigma^2 e^{-\tau} e^{-i\tau}}{2}$.

Calculul pe intervale al funcției densitate spectrală de putere.

```
Syy1=sympy.integrate(dRyy1exp*sympy.exp(-sympy.I*omega*tau),(tau,-
    oo,0))
sympy.simplify(Syy1)
```

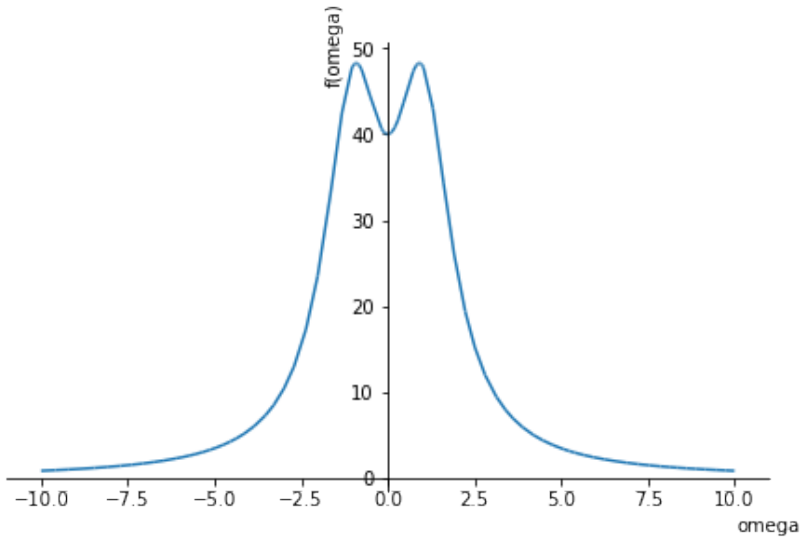
```
Syy2=sympy.integrate(dRyy2exp*sympy.exp(-sympy.I*omega*tau),(tau,0,
    oo))
sympy.simplify(Syy2)
```

```
Syy=sympy.simplify(Syy1+Syy2)
Syy
```

Rezultat: $S_{yy}(\omega) = \frac{2\sigma^2\left(\omega^2+2\right)}{\omega^4+4}$.

Reprezentare grafică.

```
dSyy=Syy.subs(sigma2,dSigma2)
sympy.plot(dSyy,(omega,-10,10))
```



Graficul funcției densitate spectrală de putere a procesului aleator din enunțul problemei 3.7; $\sigma^2 = 40$.

Calculul puterii medii a procesului aleator.

```
P=1/2/sympy.pi*sympy.integrate(Syy,(omega,-oo,oo))
P
```

Rezultat: $P_{yy} = \sigma^2$.

Calculul valorii pulsației care corespunde valorii maxime a funcției S_{yy} în intervalul $\omega > 0$.

```
dSyyDiff=sympy.diff(dSyy,omega)
sympy.simplify(dSyyDiff)
```

```
dOmegaMax=sympy.solve(dSyyDiff,omega)[0]
dOmegaMax,format(dOmegaMax.evalf(),'5.3')
```

Rezultat: $\frac{dS_{yy}(\omega)}{d\omega} = \frac{160\omega\left(\omega^4-2\omega^2\left(\omega^2+2\right)+4\right)}{\left(\omega^4+4\right)^2}$; $\omega_M = 0.910$.

4

Procese dinamice liniare

Enunțuri

Problema 4.1

Într-un experiment se constată că dacă la portul de intrare al unui sistem dinamic se aplică un puls de formă dreptunghiulară cu amplitudine unitară și durată τ:

$$u(t) = \begin{cases} 1 & \text{dacă} \quad t \in (0;\tau) \\ 0 & \text{dacă} \quad t \in (-\infty;0) \cup (\tau;\infty) \end{cases},$$

atunci, portul de ieșire al sistemului rezultă semnalul reprezentat prin expresia:

$$y(t) = \left(1 - e^{\frac{-t}{\tau}}\right) \cdot \chi(t) - \left(1 - e^{\frac{-(t-\tau)}{\tau}}\right) \cdot \chi(t-\tau).$$

Se cer următoarele.

(a) Să se calculeze expresia funcției de transfer a sistemului dinamic, $G(s)$.

(b) Să se calculeze expresia semnalului la portul de ieșire al sistemului dinamic, $w(t)$ dacă la portul de intrare se aplică semnalul rampă:

$$v(t) = \begin{cases} 0 & \text{dacă} \quad t \in (-\infty;0) \\ \frac{t}{\tau} & \text{dacă} \quad t \in [0;\infty) \end{cases}.$$

Problema 4.2

Se dă un sistem dinamic este reprezentat prin funcția de transfer care urmează:

$$G(s) = \frac{100 \cdot (s+1)}{s^2 + 10s + 100}.$$

La portul de intrare al sistemului se aplică semnalul sinusoidal:

$$u(t) = 5\sqrt{2} \cdot \sin(200t).$$

Se cere să se calculeze amplitudinea răspunsului sinusoidal, Y_M și defazajul φ dintre semnalele sinusoidale la porturile de intrare/ieșire ale sistemului.

Problema 4.3

Un sistem dinamic este reprezentat prin funcția de transfer care urmează:

$$G(s) = \frac{K_a}{(T_{f1}s + 1) \cdot (T_{f2}s + 1)}.$$

La portul de intrare al sistemului se aplică semnalul sinusoidal:

$$u(t) = \sqrt{2} \cdot \sin(\omega t).$$

Fiind date valorile numerice: $K_a = 100$, $T_{f1} = 0.2$ și $T_{f2} = 0.5$ se cere să se calculeze valoarea pulsației semnalelor, ω astfel încât valoarea amplitudinii semnalului la portul de ieșire să fie egală cu unitatea?

Problema 4.4

Se consideră un sistem dinamic cu întârziere de ordinul unu având funcția de frecvență care urmează:

$$G(j\omega) = \frac{1}{1 + T_f \cdot j\omega}.$$

La portul de intrare se aplică un semnal de tip proces aleator cu densitatea spectrală de putere:

$$S_{uu}(\omega) = \frac{1}{1 + \omega^2}.$$

Se cere să se calculeze:

(a) densitatea spectrală de putere a procesului aleator la portul de ieșire, $S_{yy}(\omega)$;

(b) puterea medie a procesului aleator la portul de intrare, σ^2_{uu} și puterea medie a procesului aleator la portul de ieșire, σ^2_{yy}.

Problema 4.5

La portul de intrare al sistemului dinamic din problema precedentă se aplică un proces aleator zgomot colorat cu funcția de autocorelație:.

$$R_{uu}(\tau) = e^{-a|\tau|}.$$

Se cere să se calculeze expresia funcției de intercorelație $R_{uy}(\tau)$ dintre procesele aleatoare la porturile de intrare/ieșire ale sistemului dinamic.
Valori numerice: $a = 1$ și $T_f = 0.5$.

Problema 4.6

Modulul funcției de frecvență al unui filtru trece-jos ideal este:

$$|H(j\omega)| = \begin{cases} H_0 & \text{dacă} \quad \omega \in [-2\pi B; 2\pi B] \\ 0 & \text{dacă} \quad \omega \in (-\infty, -2\pi B) \cup (2\pi B, +\infty) \end{cases}$$

La portul de intrare al filtrului se aplică un proces aleator zgomot alb pur cu densitatea spectrală de putere cu valoarea constantă pe tot spectrul, $S_{ee} = S_0$.
Se cere să se calculeze:

a expresia funcției de autocorelație $R_{yy}(\tau)$ a procesului aleator la portul de ieșire al filtrului;

b puterea medie de zgomot la portul de ieșire al filtrului, σ^2_{yy}.

Valori numerice: $S_0 = 10^{-5}$, $H_0 = 1$, $B = 1000$.

Problema 4.7

La portul de intrare al unui filtru de tip discret cu funcția de transfer-z:

$$H(z) = \frac{b}{z + a},$$

se aplică un proces aleator de staționar tip zgomot alb discret cu densitatea spectrală de putere $S_{ee} = 1$. Se cere să se calculeze:

(a) expresia funcției densitate spectrală de putere a procesului aleator la portul de ieșire al filtrului, $S_{yy}(\omega)$,

(b) varianța procesului aleator la portul de ieșire, σ^2_{yy}.

Valori numerice: $a = 1.25$, $b = \frac{\sqrt{3}}{2}$.

Răspunsuri

Problema 4.1:

(a) $G(s) = \frac{1}{s\tau+1}$; **(b)** $w(t) = \begin{cases} 0 & \text{dacă} \quad t < 0 \\ \frac{t}{\tau} - 1 + e^{-\frac{t}{\tau}} & \text{dacă} \quad t \geq 0 \end{cases}$.

Problema 4.2:

(a) $Y_M = 3.54$ **(b)** $\varphi = -1.53 \left[\frac{\text{rad}}{\text{s}}\right]$.

Problema 4.3:

$\omega = 37.4 \left[\frac{\text{rad}}{\text{s}}\right]$.

Problema 4.4:

(a) $S_{yy}(\omega) = \frac{1}{(\omega^2+1)(T_f^2\omega^2+1)}$; **(b)** $\sigma_{uu} = 0.5$; **(c)** $\sigma_{yy} = \frac{0.5}{T_f+1}$.

Problema 4.5:

$$R_{uy}(\tau) = \begin{cases} 0.(6)e^{-\tau} & \text{dacă} \quad \tau \in (0;\infty) \\ 2e^{\tau} - 1.(3)e^{2\tau} & \text{dacă} \quad \tau \in (-\infty;0) \end{cases}$$

Problema 4.6:

(a) $R_{yy}(\tau) = \begin{cases} \frac{1.0\cdot10^{-5}\sin(2000\pi\tau)}{\pi\tau} & \text{for } \tau \neq 0 \\ 0.02 & \text{otherwise} \end{cases}$ **(b)** $\sigma_{yy}^2 = 0.02$.

Problema 4.7:

(a) $S_{yy}(\omega) = \frac{3}{4(2.5\cos(\omega)+2.5625)}$; **(b)** $\sigma_{yy}^2 = 1.(3)$.

Rezolvări

Rezolvarea problemei 4.1

Etapele rezolvării.
(a) - Calculul funcției de transfer a sistemului dinamic
Pe baza datelor problemei se parcurg următorii pași:

- se calculează transformata Laplace a functiei-semnal la portul de intrare, $U(s) = \mathcal{L}[u(t)]$;

- se calculează transformata Laplace a functiei-semnal la portul de ieșire, $Y(s) = \mathcal{L}[y(t)]$;

Funcția de transfer a sistemului se obține cu relația:

$$G(s) = \frac{Y(s)}{U(s)}.$$

(b) - Calculul expresiei semnalului la portul de ieșire, $w(t)$ corespunzător semnalului rampă, $v(t)$.

- Se calculează transformata Laplace a semnalului rampă $v(t)$: $V(s) = \mathcal{L}[v(t)]$.

- Se calculează transformata Laplace a semnalului la portul de ieșire corespunzător semnalului rampă, $W(s)$ cu relația:

$$W(s) = G(s) \cdot V(s).$$

- Cu ajutorul transformării Laplace-inverse, se calculează expresia în domeniul timpului a semnalului la portul de ieșire, $w(t) = \mathcal{L}^{-1}[W(s)]$.

Coduri - Aplicația 4.1.
Se apelează modulele necesare.

```
import sympy
import numpy
from matplotlib import pyplot
from sympy import*
from sympy import pi
init_printing()
```

Se definesc constantele, variabilele și funcțiile.

```
t=symbols('t', real=True)#variabila timp
tau=symbols('tau',positive=True)#durata pulsului - parametrul tau
s=symbols('s')#variabila complexe s
u=Function('u')(t)#functia semnal la portul de intrare (functia
    puls), u
v=Function('v')(t)#functia semnal la portul de intrare (functia
    rampa), v
y=Function('y')(t)#functia semnal la portul de iesire (
    corespunzator functiei u), y
```

```
w=Function('w')(t)#functia semnal la portul de iesire (
    corespunzator functiei v), w
Us=Function('Us')(s)#transformata Laplace a functiei u
Vs=Function('Vs')(s)#transformata Laplace a functiei v
Ys=Function('Ys')(s)#transformata Laplace a functiei y
Ws=Function('Ws')(s)#transformata Laplace a functiei w
Gs=Function('Gs')(s)#functia de transfer a sistemului dinamic
```

Se definesc expresiile funcțiilor-semnal $u(t)$ și $y(t)$.

```
u=1*Heaviside(t)-1*Heaviside(t-tau)
y=(1-exp(-t/tau))*Heaviside(t)-(1-exp(-(t-tau)/tau))*Heaviside(t-
    tau)
u,y
```

Rezultat: $\theta(t)-\theta(t-\tau)\left(1-e^{-\frac{t}{\tau}}\right)\theta(t)-\left(1-e^{\frac{-t+\tau}{\tau}}\right)\theta(t-\tau)$

Calculul transformatelor Laplace ale funcțiilor semnal.

```
Us=sympy.laplace_transform(u,t,s)[0]#expresia transformatei Laplace
    a functiei u
Ys=sympy.laplace_transform(y,t,s)[0]#expresia transformatei Laplace
    a functiei y
Us,Ys
```

Rezultat: $U(s)=\frac{1}{s}-\frac{e^{-s\tau}}{s}$; $Y(s)=\frac{(e^{s\tau}-1)e^{-s\tau}}{s(s\tau+1)}$.

Calculul funcției de transfer a sistemului dinamic.

```
Gs=Ys/Us#expresia functiei de transfer a sistemului dinamic
simplify(Gs)
```

Rezultat: $\frac{1}{s\tau+1}$.

Calculul transformatei Laplace $W(s)$ a răspunsului corespunzător celui de-al doilea semnal de intrare.

```
v=1/tau*t*Heaviside(t)#expresia functiei-semnal v
Vs=laplace_transform(v,t,s)[0]#expresia transformatei Laplace a
    functiei-semnal v
Ws=Gs*Vs#expresia transformatei Laplace a functiei-semnal w
Ws_aux=apart(expand(simplify(Ws)),s)#simplifica si descompune in
    fractii simple expresia obtinuta la pasul anterior
Ws_aux
```

Rezultat: $W(s)=\frac{\tau}{s\tau+1}-\frac{1}{s}+\frac{1}{s^2\tau}$.

Calculul expresiei in domeniul timpului $w(t)$ a răspunsului cu transformata Laplace inversă.

```
w=inverse_laplace_transform(Ws_aux,s,t)#expresia functiei semnal w
w
```

Rezultat: $w(t)=\frac{t\theta(t)}{\tau}-\theta(t)+e^{-\frac{t}{\tau}}\theta(t)$.

Rezolvarea problemei 4.2

Etapele rezolvării.
Se va utiliza formula de calcul care urmează:

$$\begin{aligned} y(t) &= Y_M \cdot \sin(\omega t + \varphi) \\ &= U_M \cdot |G(j\omega)| \cdot \sin[\omega t + \arg\{G(j\omega)\}]. \end{aligned}$$

Prin urmare, este necesar să calculăm modulul și argumentul funcției de frecvență a sistemului dinamic.
Se parcurg pașii care urmează.

- Pe baza expresiei funcției de transfer, $G(s)$ se determină expresia funcției de frecvență prin înlocuirea variabilei complexe s cu variabila $j\omega$.
- Funcția de frecvență se scrie sub forma algebrică, $G(j\omega) = \mathrm{Re}\{G(j\omega)\} + j \cdot \mathrm{Im}\{G(j\omega)\}$ și se pun în evidență părțile reală și imaginară ale acesteia.
- Se calculează modulul și argumentul funcției de frecvență.

$$|G(j\omega)| = \sqrt{[\mathrm{Re}\{G(j\omega)\}]^2 + [\mathrm{Im}\{G(j\omega)\}]^2},$$

$$\arg\{G(j\omega)\} = \arctan\left[\frac{\mathrm{Im}\{G(j\omega)\}}{\mathrm{Re}\{G(j\omega)\}}\right]$$

- Rezultă amplitudinea semnalului sinusoidal la portul de ieșire, Y_M și defazajul dintre semnalele la cele două porturi, φ astfel:

$$Y_M = U_M \cdot |G(j\omega)|; \quad \varphi = \arg\{G(j\omega)\}.$$

Coduri - Aplicația 4.2.
Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
from sympy.matrices import Matrix
from matplotlib import pyplot
sympy.init_printing()
```

Definitiile variabilelor și funcțiilor

```
t=symbols('t',real=True)#variabila timp
s=symbols('s')#variabila complexa s
omega=symbols('omega',positive=True)#variabila omega - pulsatia
    semnalului sinusoidal
U=symbols('U', real=True)#amplitudinea semnalului sinusoidal
Gs=sympy.Function('Gs')(s)#functia de transfer
G=sympy.Function('G')(omega)#functia de frecventa
u=sympy.Function('u')(t)#semnalul la portul de intrare
y=sympy.Function('y')(t)#raspunsul sinusoidal al sistemului dinamic
```

Expresiile funcției de transfer și funcției de frecvență

```
Gs=100*(s+1)/(s**2+10*s+100)#expresia functiei de transfer
G=Gs.subs(s,I*omega)#expresia functiei de frecventa (fdf)
G
```

Rezultat: $\frac{100i\omega+100}{-\omega^2+10i\omega+100}$.

Expresiile părților reală și imaginară ale funcției de frecvență.

```
G_re=simplify(re(G))#partea reala a fdf
G_im=simplify(im(G))#partea imaginara a fdf
G_re,G_im
```

Rezultat: $\frac{100\left(9\omega^2+100\right)}{\omega^4-100\omega^2+10000}$; $\frac{100\omega\left(90-\omega^2\right)}{100\omega^2+\left(\omega^2-100\right)^2}$.

Modulul și argumentul funcției de frecvență.

```
G_mod=simplify(Abs(G))#modulul fdf
G_arg=simplify(atan(G_im/G_re))#argumentul fdf
G_mod,G_arg
```

Rezultat: $\frac{100\sqrt{\omega^2+1}}{\sqrt{\omega^4-100\omega^2+10000}}$; $-\operatorname{atan}\left(\frac{\omega\left(\omega^2-90\right)}{9\omega^2+100}\right)$.

Expresia răspunsului sinusoidal al sistemului dinamic.

```
u=U*sin(omega*t)#expresia semnalului la portul de intrare
y=U*G_mod*sin(omega*t+G_arg)#expresia raspunsului
simplify(y)
```

Rezultat: $\frac{100U\sqrt{\omega^2+1}\sin\left(\omega t-\operatorname{atan}\left(\frac{\omega\left(\omega^2-90\right)}{9\omega^2+100}\right)\right)}{\sqrt{\omega^4-100\omega^2+10000}}$.

Calcule numerice.

```
dU=5*sqrt(2)#amplitudinea semnalului sinusoidal la portul de
    intrare
dOmega1=200#pulsatia semnalelor sinusoidale
du=u.subs(U,dU).subs(omega,dOmega1)#expresia semnalului la portul
    de intrare cu valori numerice
dy=y.subs(U,dU).subs(omega,dOmega1)#expresia raspunsului sinusoidal
     cu valori numerice
dy
```

Rezultat: $\frac{5\sqrt{75552658}\sin\left(200t-\operatorname{atan}\left(\frac{6140}{277}\right)\right)}{12277}$.

```
dY_M=dy.args[0]*dy.args[1]#din expresia precedenta, preia factorii
    modulului
dYM=dY_M.evalf()#variabila auxiliara
format(dYM,'5.3')
```

Rezultat: $Y_M = 3.54$.

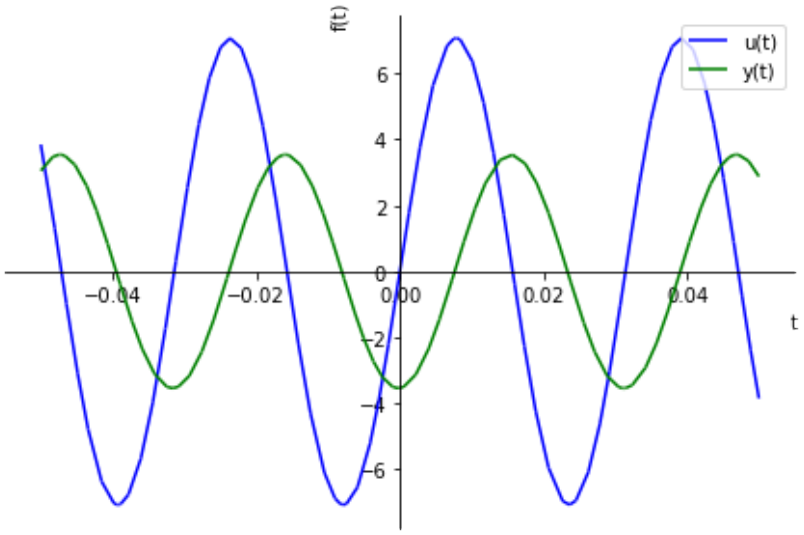
```
dY_Arg=dy.args[2].args[0].subs(t,0).evalf()#din expresia precedenta
    , preia factorii argumentului
format(dY_Arg,'5.3')
```

Rezultat: $\varphi = -1.53$.

Reprezentare grafică.

```
p1=plot(du,(t,-0.05,0.05),show=False,line_color='blue', label='u(t)
    ',legend=True)
p2=plot(dy,(t,-0.05,0.05),show=False,line_color='green',label='y(t)
    ')
p1.append(p2[0])
p1.show()
```

Rezultat:



Graficele semnalelor la porturile sistemului; $u(t)$ - semnalul la portul de intrare și $y(t)$ - semnalul la portul de ieșire.

Rezolvarea problemei 4.3

Etapele rezolvării.

Se folosește aceeași metodă de calcul ca și la rezolvarea problemei 4.2.
Se scrie expresia amplitudinii răspunsului sinusoidal al sistemului dinamic:

$$Y_M = U_M \cdot |G(j\omega)|.$$

Cerința din enunțul problemei este echivalentă cu ecuația:

$$U_M \cdot |G(j\omega)| = 1,$$

în care necunoscuta este pulsația ω.

Coduri - Aplicația 4.3.
Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
from sympy.matrices import Matrix
from matplotlib import pyplot
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor

```
K_a=sympy.symbols('K_a',positive=True)#factorul de amplificare
T_f1=sympy.symbols('T_f1',positive=True)#constanta de timp T_f1
T_f2=sympy.symbols('T_f2',positive=True)#constanta de timp T_f2
s=sympy.symbols('s')#variabila complexa s
omega=symbols('omega',real=True)# pulsatia
Gs=sympy.Function('Gs')(s)#functia de transfer
G=sympy.Function('G')(omega)#functia de frecventa
```

Expresiile funcției de transfer și funcției de frecvență.

```
Gs=K_a/((T_f1*s+1)*(T_f2*s+1))#expresia functiei de transfer
Gs
```

Rezultat: $\frac{K_a}{(T_{f1}s+1)(T_{f2}s+1)}$.

```
Gf=Gs.subs(s,I*omega)#expresia functiei de frecventa
Gf
```

Rezultat: $\frac{K_a}{(iT_{f1}\omega+1)(iT_{f2}\omega+1)}$.

Calculul modulului funcției de frecvență.

```
Gf_modul=simplify(Abs(Gf))
Gf_modul
```

Rezultat: $\frac{K_a}{\sqrt{T_{f1}^2\omega^2+1}\sqrt{T_{f2}^2\omega^2+1}}$.

Calcule cu valori numerice.

```
dK_a=100#factorul de proportionalitate
dT_f1=0.2#constanta de timp
dT_f2=0.5#constanta de timp
dGf=Gf.subs(K_a,dK_a).subs(T_f1,dT_f1).subs(T_f2,dT_f2)#expresia
    functiei de frecventa cu valori numerice
dGf
```

Rezultat: $\frac{100}{(0.2i\omega+1)(0.5i\omega+1)}$.

```
dGf_modul=simplify(Abs(dGf))#expresia modulului functiei de
    frecventa cu valori numerice
dGf_modul
```

Rezultat: $\frac{100}{\sqrt{0.01\omega^4+0.29\omega^2+1}}$.

```
dOmega=sympy.solve(dY-1,omega)#Calculul valorii pulsatiei pentru
    care amplitudinea semnalului de iesire este unitara
dOmega_aux=dOmega[1].evalf()#variabila auxiliara
format(dOmega_aux,'5.3')
```

Rezultat: $\omega = 37.4$.

Calculul de verificare.

```
dY1=dY.subs(omega,dOmega[1])
dY1.evalf()
```

Rezultat: 1.0.

Rezolvarea problemei 4.4

Etapele rezolvării.

(a) - Calculul densității spectrale de putere a procesului aleator la portul de ieșire

Calculul se face cu formula care urmează.

$$S_{yy}(\omega) = |G(j\omega)|^2 \cdot S_{uu}(\omega).$$

(b) - Calcul puterilor medii ale proceselor aleatoare la porturile de intrare/ieșire.

Se integrează densitățile spectrale de putere pe tot domeniul de variație al pulsației:

$$\sigma_{uu}^2 = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} S_{uu}(\omega)d\omega; \quad \sigma_{yy}^2 = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} S_{yy}(\omega)d\omega.$$

.

Coduri - Aplicația 4.4.

Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
T_f=sympy.symbols('T_f',positive=True)#constanta de timp T_f
omega=sympy.symbols('omega',real=True)# pulsatia
G=sympy.Function('G')(omega)#functia de frecventa
G_modul=sympy.Function('G_modul')(omega)#modulul functiei de
    frecventa
S_uu=sympy.Function('S_uu')(omega)#functia densitate spectrala de
    putere a procesului aleator la portul de intrare
S_yy=sympy.Function('S_yy')(omega)#functia densitate spectrala de
    putere a procesului aleator la portul de iesire
sigma2_uu=sympy.symbols('sigma2_uu',positive=True)#varianta
    procesului aleator la portul de intrare
sigma2_yy=sympy.symbols('sigma2_yy',positive=True)#varianta
    procesului aleator la portul de iesire
```

Expresiile funcțiilor.

```
S_uu=1/(1+omega**2)#functia densitate spectrala de putere
G=1/(1+T_f*sympy.I*omega)#functia de frecventa a sistemului dinamic
S_uu,G
```

Rezultat: $\frac{1}{\omega^2+1}$; $\frac{1}{iT_f\omega+1}$.

Calculul modulului funcției de frecvență.

```
G_modul=sympy.Abs(G)
sympy.expand(G_modul)
```

Rezultat: $\frac{1}{\sqrt{T_f^2\omega^2+1}}$.

Calculul densității spectrale de putere a procesului aleator (PA) la portul de ieșire.

```
S_yy=G_modul**2*S_uu
S_yy
```

Rezultat: $\frac{1}{(\omega^2+1)\left(T_f^2\omega^2+1\right)}$.

Calculul puterilor medii ale proceselor aleatoare la porturile de intrare respectiv ieșire.

```
sigma2_uu=1/2/sympy.pi*sympy.integrate(S_uu,(omega,-oo,oo))
sigma2_uu
```

Rezultat: 0.5.

```
sigma2_yy=1/2/sympy.pi*sympy.integrate(S_yy,(omega,-oo,oo))
sympy.simplify(sigma2_yy)
```

Rezultat: $\frac{0.5}{T_f+1}$.

Rezolvarea problemei 4.5

Etapele rezolvării.

Se calculează funcția pondere a sistemului dinamic cu transformarea Laplace inversă:

$$g(\xi) = \mathcal{L}^{-1}G(s) = \frac{1}{T_f} \cdot e^{\frac{-\xi}{T_f}} \cdot \chi(\xi).$$

Cunoscând expresiile funcției pondere, g și funcției de autocorelație a procesului aleator la portul de intrare, R_{uu}, funcția de intercorelație dintre procesele aleatoare la porturile de intrare/ieșire, R_{uy} se calculează cu formula:

$$R_{uy}(\tau) = \int_{-\infty}^{\infty} g(\xi) \cdot R_{uu}(\tau + \xi)d\xi.$$

În integrala de mai sus, integrarea se face în raport cu variabila $\xi \in \mathbb{R}$ și variabila $\tau \in \mathbb{R}$ este parametru. Deoarece valorile funcției pondere sunt egale cu zero în intervalul $(-\infty; 0)$ și, ținând cont de expresiei funcției $R_u u(\tau)$ din enunțul problemei, calculul se face după cum urmează:

$$R_{uy}(\tau) = \int_{0}^{\infty} g(\xi) \cdot R_{uu}(\tau + \xi)d\xi = \int_{0}^{\infty} \frac{1}{T_f} e^{\frac{-\xi}{T_f}} \chi(\xi) \cdot e^{-a|\tau+\xi|}d\xi.$$

Calcul integralei se face pe trei intervale: (a) $\tau > 0$ $\xi > 0$; (b) $\tau < 0$ și $(\xi + \tau) > 0$ respectiv, (c) $\tau < 0$ și $(\xi + \tau) < 0$.

Coduri - Aplicația 4.5.
Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
a=sympy.symbols('a', positive=True)#parametrul a
T_f=sympy.symbols('T_f',positive=True)#constanta de timp T_f
s=sympy.symbols('s',real=True)#variabila complexa s
tau=sympy.symbols('tau', real=True)#variabila tau - decalaj in timp
xi=sympy.symbols('xi',positive=True)#variabila xi - decalaj in timp
Gs=sympy.Function('Gs')(s)#functia de transfer a sistemului dinamic
g=sympy.Function('g')(xi)#functia pondere a sistemului dinamic
R_uu=sympy.Function('R_uu')(tau)#functia de autocorelatie a
    procesului aleator (PA) la portul de intrare
```

Expresia funcției pondere a sistemului dinamic.

```
Gs=1/(T_f*s+1)#expresia functiei de transfer a sistemului dinamic
g=inverse_laplace_transform(Gs,s,xi)#expresia functiei pondere
g
```

Rezultat: $\frac{e^{-\frac{\xi}{T_f}}}{T_f}$.

Calculul funcției de intercorelație în intervalul $\tau > 0$ și $(\tau + \xi) > 0$.

```
g=1/T_f*exp(-xi/T_f)*sympy.Heaviside(xi)#expresia functiei pondere
R1uu=sympy.exp(-a*(tau+xi))#expresia functiei de autocorelatie a PA
    la portul de intrare
R1uy=integrate(g*R1uu,(xi,0,oo))#calculul expresiei functiei de
    intercorelatie
R1uy
```

Rezultat: $\frac{e^{-a\tau}}{T_f a+1}$.

Calculul funcției de intercorelație in intervalul $\tau < 0$ și $(\tau + \xi) < 0$.

```
R2uu=sympy.exp(a*(tau+xi))#expresia functiei de autocorelatie a PA
    la portul de intrare
R2uy=integrate(g*R2uu,(xi,0,-tau))#calculul expresiei functiei de
    intercorelatie
sympy.simplify(R2uy)
```

Rezultat:

$$\begin{cases} \frac{1}{T_f a e^{-\frac{\tau}{T_f}} - e^{-\frac{\tau}{T_f}}} - \frac{e^{a\tau}}{T_f a - 1} & \text{for } a \neq \frac{1}{T_f} \\ -\frac{\tau e^{\frac{\tau}{T_f}}}{T_f} & \text{otherwise} \end{cases}.$$

Calculul funcției de intercorelație in intervalul $\tau < 0$ și $(\tau + \xi) > 0$.

```
R3uu=sympy.exp(-a*(tau+xi))#expresia functiei de autocorelatie a PA
    la portul de intrare
R3uy=integrate(g*R3uu,(xi,-tau,oo))#calculul expresiei functiei de
    intercorelatie
sympy.simplify(R3uy)
```

Rezultat: $\frac{e^{\frac{\tau}{T_f}}}{T_f a+1}$.

Expresia funcției de intercorelație in intervalul $\tau < 0$ pe ambele intervale.

```
Ruy=R2uy+R3uy#expresia functiei de intercorelatie in intervalul (-
    oo; 0)
Ruy
```

Rezultat:

$$\begin{cases} \frac{1}{T_f a e^{-\frac{\tau}{T_f}} - e^{-\frac{\tau}{T_f}}} - \frac{e^{a\tau}}{T_f a - 1} & \text{for } a \neq \frac{1}{T_f} \\ -\frac{\tau e^{\frac{\tau}{T_f}}}{T_f} & \text{otherwise} \end{cases} + \frac{e^{\frac{\tau}{T_f}}}{T_f a+1}.$$

Calcule numerice: $a = 1$ și $T_f = 0.5$.

```
dA=1#valoarea numerica a parametrului a
dTf=0.5#valoarea numerica a parametrului T_f
dR1uy=R1uy.subs(a,dA).subs(T_f,dTf)#valoarea numerica a functiei de
    intercorelatie in intervalul tau > 0 si tau + xi > 0
dR2uy=R2uy.subs(a,dA).subs(T_f,dTf)#valoarea numerica a functiei de
    intercorelatie in intervalul tau < 0 si tau + xi < 0
dR3uy=R3uy.subs(a,dA).subs(T_f,dTf)#valoarea numerica a functiei de
    intercorelatie in intervalul tau < 0 si tau +xi > 0
dR23uy=dR2uy+dR3uy
dR1uy,dR23uy
```

Rezultat:

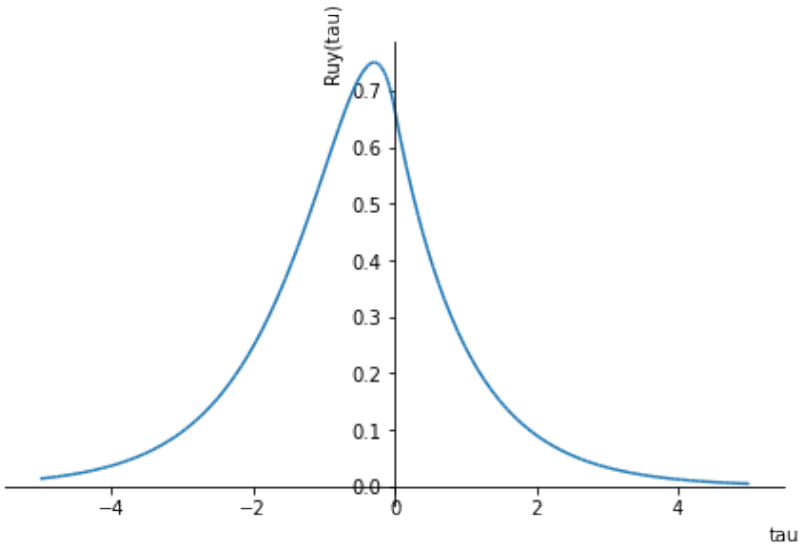
$0.(6)e^{-\tau}$ pentru $\tau > 0$

$2.0e^{\tau} - 1.(3)e^{2.0\tau}$ pentru $\tau < 0$

Reprezentare grafică.

```
p1=sympy.plot(dR1uy,(tau,0,5),show=False,  xlabel='tau',ylabel='Ruy(
    tau)')
p2=sympy.plot(dR23uy,(tau,-5,0),show=False)
p1.append(p2[0])
p1.show()
```

Rezultat:



Graficul funcției de intercorelație $R_{uy}(\tau)$ pentru procesele aleatoare din problema 4.5.

Rezolvarea problemei 4.6

Etapele rezolvării.

(a) - Calculul funcției de autocorelație a procesului aleator la portul de ieșire al filtrului.
Densitatea spectrală de putere la portul de ieșire al filtrului este dată de relația:

$$S_{yy}(\omega) = |H(j\omega)|^2 S_{uu}(\omega).$$

Funcția de autocorelație a procesului aleator este transformata Fourier inversă a funcției densitate spectrală de putere:

$$R_{yy}(\tau) = \mathcal{F}^{-1}\left[S_{yy}(\omega)\right] = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{yy}(\omega)e^{j\omega\tau}d\omega.$$

Pe baza datelor din enunțul problemei, rezultă că trebuie calculată următoarea integrală.

$$R_{yy}(\tau) = \frac{1}{2\pi}\int_{-2\pi B}^{2\pi B} H_0^2 S_0 e^{j\omega\tau}d\omega.$$

(b) - Calculul puterii medii de zgomot la portul de ieșire al filtrului.
Puterea medie de zgomot este egală cu valoarea funcției de autocorelație pentru $\tau = 0$, respectiv: $\sigma_{yy}^2 = R_{yy}(0)$.

Coduri - Aplicația 4.6.
Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor simbolice.

```
tau=sympy.symbols('tau', real=True)#variabila tau - decalaj in timp
omega=sympy.symbols('omega',real=True)#variabila pulsatie extinsa
    pe axa reala
B=sympy.symbols('B', positive=True)#largimea de banda a filtrului
H_0=sympy.symbols('H_0',positive=True)#valoarea modulului functiei
    de frecventa a filtrului
S_ee=sympy.symbols('S_ee',positive=True)#valoarea densitatii
    spectrale de putere a procesului aleator la portul de intrare
R_yy=sympy.Function('R_yy')(tau)#functia de autocorelatie a
    procesului aleator (PA) la portul de iesire
```

Expresia funcției de autocorelație la portul de ieșire al filtrului.

```
R_yy=1/2/sympy.pi*sympy.integrate(H_0**2*S_ee*exp(sympy.I*omega*tau
    ),(omega,-B,B))
sympy.simplify(R_yy)
```

Rezultat: $R_{yy}(\omega) = \begin{cases} \frac{1.0H_0^2 S_{ee}\sin(B\tau)}{\pi\tau} & \text{for } \tau \neq 0 \\ \frac{1.0BH_0^2 S_{ee}}{\pi} & \text{otherwise} \end{cases}$.

Calculul puterii medii de zgomot a filtrului.

```
sigma2_yy=R_yy.subs(tau,0)
sigma2_yy
```

Rezultat: $\frac{1.0BH_0^2S_0}{\pi}$.

Calcule numerice.

```
dSee=sympy.Pow(10,-5)#valoarea densitatii spectrale de putere a
    procesului aleator (PA) la portul de intrare
dB=2*sympy.pi*1000#valoarea largimii de banda a filtrului
dH0=1#valoarea modulului functiei de frecventa a filtrului
dRyy=R_yy.subs(S_ee,dSee).subs(B,dB).subs(H_0,dH0)#expresia
    functiei de autocorelatie a PA la portul de iesire
sympy.simplify(dRyy)
```

Rezultat: $R_{yy}(\omega) = \begin{cases} \frac{1.0 \cdot 10^{-5} \sin(2000\pi\tau)}{\pi\tau} & \text{for } \tau \neq 0 \\ 0.02 & \text{otherwise} \end{cases}$.

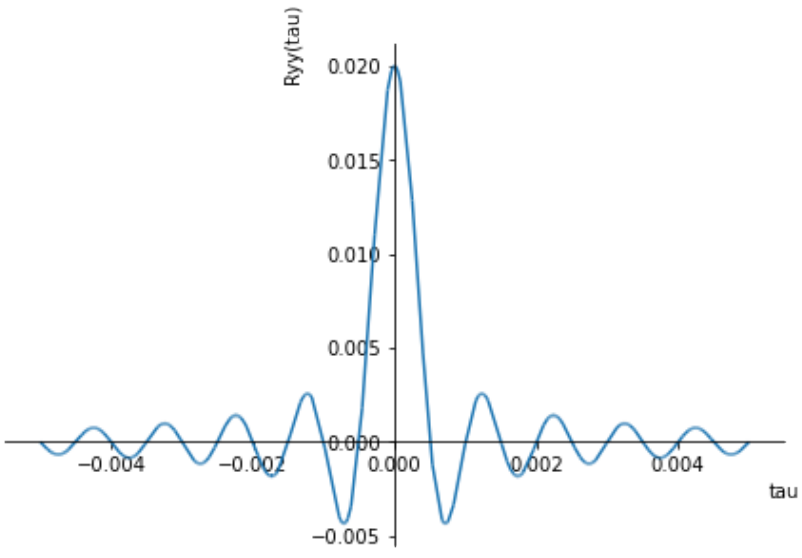
```
dSigma2yy=sigma2_yy.subs(S_ee,dSee).subs(B,dB).subs(H_0,dH0)#
    valoarea puterii medii de zgomot a PA la portul de iesire
dSigma2yy
```

Rezultat: $\sigma 2_y y = 0.02$.

Reprezentare grafică.

```
p1=sympy.plot(dRyy,(tau,-0.005,0.005),show=False, xlabel='tau',
    ylabel='Ryy(tau)')
p1.show()
```

Rezultat:



Reprezentarea grafică a funcției de autocorelație a procesului aleator la portul de ieșire al filtrului din problema 4.6.

Rezolvarea problemei 4.7

Etapele rezolvării.

(a) Calculul densității spectrale de putere a procesului aleator la portul de ieșire al filtrului.
Se parcurg următorii pași.

- Se calculează expresia funcției de frecvență pe cercul unitate și expresia modulului la pătrat al acesteia.

$$H(e^{j\omega}) = H(z)|_{z=e^{j\omega}}; \quad \left|H(e^{j\omega})\right|^2 = H(e^{j\omega}) \cdot H(e^{-j\omega}).$$

- Expresia densității spectrale de putere a procesului aleator la portul de ieșire al filtrului este:

$$S_{yy}(\omega) = \left|H(e^{j\omega})\right|^2 \cdot S_{ee}(\omega).$$

(b) Varianța procesului aleator la portul de ieșire.
Se calculează cu formula:

$$\sigma_{yy}^2 = \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} S_{yy}(\omega) d\omega.$$

Coduri - Aplicația 4.7.
Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
a=sympy.symbols('a', real=True)#definitia constantei a
b=sympy.symbols('b',real=True)#definitia constantei b
z=sympy.symbols('z')#definitia variabilei complexe z
omega=sympy.symbols('omega', real=True)#definitia variabilei omega
    - pulsatia
Hz=sympy.Function('Hz')(z)#defintia functiei de transfer-z
Hf=sympy.Function('Hf')(omega)#definitia functiei de frecventa pe
    cercul unitate in planul z
S_uu=sympy.Function('S_uu')(omega)#definitia functiei densitate
    spectrala de putere a PA la portul de intrare
S_yy=sympy.Function('S_yy')(omega)#definitia functiei densitate
    spectrala de putere a PA la portul de iesire
```

Expresia funcției de transfer-z

```
Hz=b/(z+a)#functia de transfer-z
Hz
```

Rezultat: $\frac{b}{a+z}$.
Expresia funcției de frecvență pe cercul unitate din planul-z.

```
Hf=Hz.subs(z,sympy.exp(sympy.I*omega))#functia de frecventa
Hf2_modul=Hz.subs(z,sympy.exp(-sympy.I*omega))*Hz.subs(z,sympy.exp(
    sympy.I*omega))#modulul functiei de frecventa
Hf2_modul1=sympy.simplify(Hf2_modul.rewrite(sin))#expresia
    modulului cu formula lui Euler
Hf, Hf2_modul, Hf2_modul1
```

Rezultat: $|H(e^{j\omega})|^2 = \frac{b^2}{a^2+2a\cos(\omega)+1}$.

OBSERVAȚIE.
Funcția $|H(e^{j\omega})|^2$ conține un termen în $\cos(\omega)$.
Pentru calculul integralei care apare în expresia variației $\sigma^2_{yy}(\omega)$, se face substituția $t = \tan\left(\frac{\omega}{2}\right)$.
Rezultă: $\omega = 2\,\text{atan}(t)$, $d\omega = \frac{2}{t^2+1}dt$.
Noile limite de integrare vor fi $t \to -\infty$ și respectiv $t \to \infty$.

Definiția și expresia variabilei $t = \tan\frac{\omega}{2}$.

```
t=sympy.symbols('t',real=True)#defintia variabilei t
dOmega=2*sympy.atan(t)#relatia dintre cele doua variabile
dDiffOmega=diff(dOmega)#diferentiala variabilei omega
dOmega, dDiffOmega
```

Rezultat: $\omega = 2\,\text{atan}\,(t)$, $d\omega = \frac{2}{t^2+1}dt$.

Calculul funcției densitate spectrală de putere a procesului aleator la portul de ieșire.

```
Hf2_modul2=sympy.simplify(Hf2_modul1.subs(cos(omega),(1-t**2)/(1+t
    **2)))
Hf2_modul2
```

Rezultat: $\frac{b^2\left(t^2+1\right)}{-2a(t^2-1)+(a^2+1)(t^2+1)}$.

Calcule cu valori numerice.

```
dA=1.25#valoarea constantei a
dB=sympy.sqrt(3)/2#valoarea constantei b
dHf2_modul1=Hf2_modul1.subs(a,dA).subs(b,dB)#Hf_modul1 cu valori
    numerice
dHf2_modul2=Hf2_modul2.subs(a,dA).subs(b,dB)#Hf_modul2 cu valori
    numerice
S_uu=1#densitatea spectrala de putere a PA la portul de intrare
S_yy=dHf2_modul1*S_uu#densitatea spectrala de putere a PA la portul
     de iesire
PrimInt=1/2/sympy.pi*sympy.integrate(dHf2_modul2*S_uu*dDiffOmega,t)
   #primitiva integralei definite
sigma2_yy=limit(PrimInt,t,oo)-limit(PrimInt,t,-oo)#integrala
    definita - varianta
S_yy, sigma2_yy
```

Rezultat: $S_{yy}(\omega) = \frac{3}{4(2.5\cos(\omega)+2.5625)}$; $\sigma^2_{yy} = 1.(3)$.

5

Metode neparametrice

Enunțuri

Problema 5.1

Într-un experiment de identificare a fost determinată experimental constanta unui resort elastic. Experimentul a constat în observarea valorilor elongației (alungirii) resortului, y, suspendat vertical și supus acțiunii forței de greutate, u a unor greutăți etalonate amplasate pe un taler montat în partea de jos a acestuia.
Linia de măsurare a elongației era perturbată de zgomot cu media diferită de zero și, drept urmare, datele prelevate din experiment au rezultat cu perturbații.
Pentru a obține cea mai bună aproximare a constantei resortului, analiza datelor prelevate s-a bazat pe următorul model:

$$y(u) = a \cdot u + b,$$

în care: a este valoarea adevărată a constantei resortului și b este abaterea de măsurare (bias) a traductorului.

Datele prelevate în cadrul experimentului sunt prezentate sub formă numerică în tabelele 5.1 și 5.2 precum și sub formă grafică în figura 5.1.

Tabela 5.1: Valorile greutății - aU.

0.000 0.500 1.000 1.500 2.000 2.500 3.000 3.500 4.000 4.500 5.000 5.500 6.000 6.500 7.000 7.500 8.000 8.500 9.000 9.500

Tabela 5.2: Valorile elongației - aY.

0.022 0.120 0.223 0.263 0.301 0.544 0.596 0.670 0.723 0.912 0.963 1.129 1.274 1.419 1.468 1.537 1.583 1.726 1.863 1.963

Folosind datele experimentale prelevate, se cere să se estimeze cea mai bună aproximare liniară a constantei de elasticitate a resortului, $\hat{a}$ precum și cea mai bună aproximare a bias-ului traductorului, $\hat{b}$.

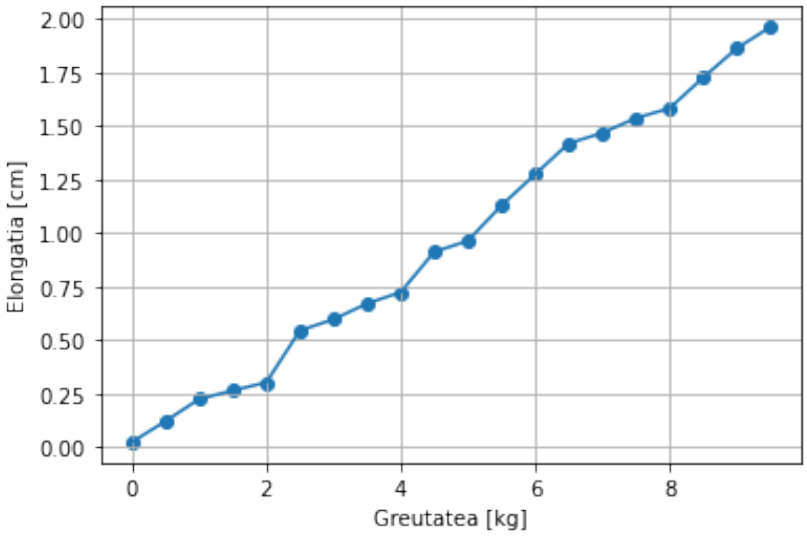


Figura 5.1: Reprezentarea grafică a datelor prelevate în experimentul din problema 5.1.

Problema 5.2

Fiind dat un sistem dinamic de tip proporțional cu întârziere de ordinul unu, $PT1$.

(a) Să se demonstreze că răspunsul liber cu condiție inițială diferită de zero, $y(0) = Y_0$ al sistemului dinamic se reprezintă printr-o expresie asemănătoare cu funcția pondere a acestuia; ce parametru poate fi estimat cu ajutorul răspunsului liber al acestei clase de procese.

(b) Fiind date două valori $y(t_1)$ și $y(t_2)$ pe curba răspunsului liber între care există relația $\frac{y(t_1)}{y(t_2)} = e$. Să se arate că diferența valorilor momentelor pe axa timpului, $t_2 - t_1$ este numeric egală cu constanta de timp T_f a procesului.

(c) Să se arate că tangenta la graficul curbei răspunsului liber în punctul de adscisă 0_+ intersectează axa orizontală în punctul de abscisă T_f.

Problema 5.3

Un inginer și-a propus să studieze procesul curgerii apei printr-o instalație industrială formată dintr-un rezervor cu suprafață liberă și un sistem de evacuare - electrovalvă și circuit de golire - amplasat la partea inferioară a rezervorului. În acest scop, inginerul a observat variația înălțimii coloanei de apă din rezervor pe durata golirii fără realimentare a rezervorului. Datele experimentale sunt prezentate în format numeric în tabelele (5.3), (5.4) și în format grafic în figura (5.2).

Tabela 5.3: Valorile variabilei timp - aTimp [min].

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0 20.5 21.0 21.5 22.0 22.5 23.0 23.5 24.0 24.5

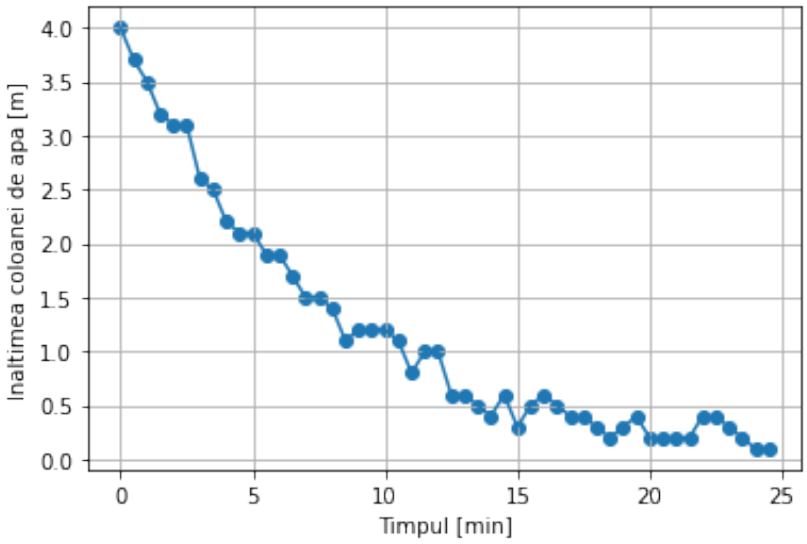


Figura 5.2: Variația înălțimii coloanei de apă din rezervor în funcție de timp determinată în experimentul din problema 5.3.

Tabela 5.4: Valorile înălțimii coloanei de apă - aY [m].

4.0 3.9 3.5 3.3 3.2 2.9 2.5 2.7 2.4 2.4 1.9 1.9 1.8 1.6 1.6 1.5 1.6 1.2 1.2 1.1 0.9 1.0 1.0 1.0 0.7 0.8 0.6 0.7 0.8 0.7 0.4 0.5 0.5 0.3 0.5 0.3 0.4 0.4 0.2 0.2 0.4 0.2 0.4 0.2 0.2 0.1 0.3 0.3 0.2 0.1

Bazat pe aceste rezultate, inginerul a formulat ipoteza ca procesul analizat poate fi aproximat sub forma unui proces de tip proporțional cu întârziere de ordinu unu PT_1 Pentru a elimina efectele erorilor de măsurare, inginerul a decis să aproximeze curba valorilor determinate experimental cu o parabolă de ecuație:

$$\hat{y}(t) = \hat{a} \cdot t^2 + \hat{b} \cdot t + \hat{c}.$$

Se cer următoarele.

(a) Să se estimeze valorile parametrilor curbei de aproximare, $\hat{a}$, $\hat{b}$ și $\hat{c}$.

(b) Cu ajutorul rezultatelor obținute la problema precedentă, să se estimeze valoarea constantei de timp a procesului studiat.

Problema 5.4

(a) Să se arate că funcția pondere a unui sistem dinamic cu întârziere de ordinul unu, $PT1$ având factorul de proporționalitate K_a și constanta de timp T_f verifică relația:

$$g(t+\tau) = g(t) \cdot e^{\frac{-\tau}{T_f}}, \forall t \in \mathbb{R}_+; \quad \forall \tau \in \mathbb{R}_+,$$

în care t reprezintă timpul și τ reprezintă deplasarea în domeniul timpului.

(b) Fiind dată o valoare a deplasării τ, care este semnificația factorului $a = e^{\frac{-\tau}{T_f}}$ în planul $(g(t)Og(t+\tau))$?

(c) Să se arate că și răspunsul liber cu condiție inițială nenulă al sistemelor $PT1$ verifică o relație asemănătoare.

Problema 5.5

Pentru acordarea regulatoarelor sistemului de reglare automată al unei turbine eoliene trebuie cunoscută valoarea constantei de timp a rotorului acesteia.
Estimarea constantei de timp se face cu metoda autofrânării și ipoteza de procesul frânării libere este un proces de tip $PT1$.
Metoda autofrânării constă în observarea variației vitezei unghiulare a rotorului în funcție de timp, pe durata frânării libere a acestuia de la viteza unghiulară nominală la zero sub acțiunea forțelor de frecare mecanice și aerodinamice.
Dacă datele prelevate sunt puternic perturbate de zgomot aditiv atunci analiza datelor experimentale se face cu metoda care urmează:

- se alege o valoare a deplasării în timp τ și se reprezintă grafic dependența $y(t+\tau) = f[y(t)]$;
- cu ajutorul metodei celor mai mici pătrate se estimează parametrii dreptei de regresie, $\hat{a}$ și $\hat{b}$ în planul $(y(t)Oy(t+\tau))$;
- se estimează constanta de tip din relația $\hat{a} = e^{\frac{-\tau}{T_f}}$.

Se cere să se estimeze valoarea numerică a constantei de timp a rotorului turbinei eoliene T_f bazat pe datele experimentale prezentate grafic în figura () și numeric tabelele (5.5) și (5.6) .

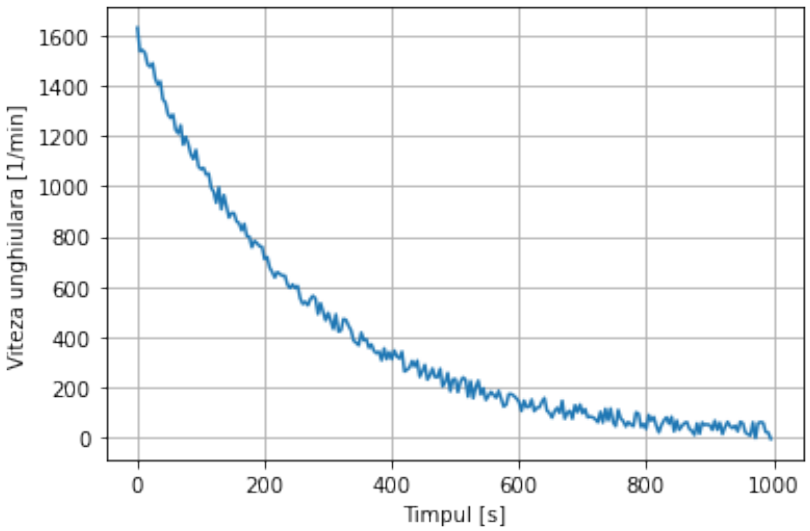


Figura 5.3: Dependența vitezei unghiulare în funcție de timp la frânarea liberă a rotorului turbinei eoliene din problema 5.5.

Problema 5.6

Un sistem dinamic de tip $PT2$ supra-amortizat are funcția de transfer:

$$G(s) = \frac{K_a}{(T_{f1}s+1)\cdot(T_{f2}s+1)}; \quad , T_{f1}, T_{f2} \in \mathbb{R}_+.$$

Se cer următoarele:

(a) să se determine expresia funcției pondere, $g(t)$;

(b) fiind dată o deplasare în timp τ, se notează cu $a_1 = e^{\frac{-\tau}{T_{f1}}}$ și $a_2 = e^{\frac{-\tau}{T_{f2}}}$; să se arate că funcția pondere a sistemului dinamic verifică relația:

$$g(t+2\tau) - (a_1 + a_2) \cdot g(t+\tau) + a_1 \cdot a_2 \cdot g(t) = 0.$$

Problema 5.7

Măsurarea modulului de elasticitate al pielii umane se realizează cu un dispozitiv compus dintr-un electromagnet cu plujer care acționează poansonul aflat în contact cu pielea. Dispozitivul este prevăzut cu o linie de măsurare cu care se observă deplasarea poansonului pe durata testării și un modul de calcul cu care se estimează modulul de elasticitate.
Modelul procesului dinamic care are loc pe durata testării poate fi aproximat printr-un proces cu întârziere de ordinul doi supra-amortizat reprezentat prin funcția de transfer din problema precedentă.
În continuare, a fost efectuat un test de măsurare a constantei de elasticitate a pielii rezultând datele prezentate grafic în figura (5.4) și numeric în tabelele (5.7).

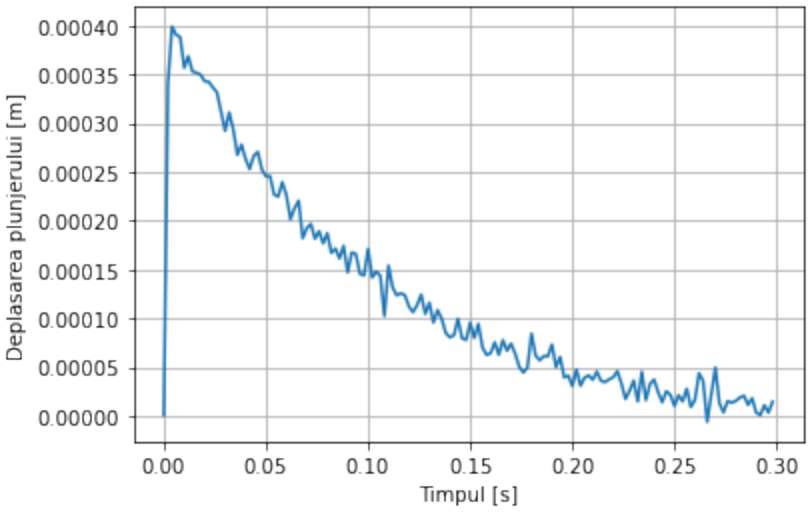


Figura 5.4: Dependența deplasării plunjerului în funcție de timp în experimentul din problema 5.7.

Analiza datelor experimentale se face astfel:

- se folosește rezultatul din problema precedentă;
- se alege o deplasare în domeniul timpului τ, se calculează noile variabile $\zeta = g(t+2\tau)$, $\gamma = g(t+\tau)$ și $\xi = g(t)$;
- folosind noile variabile, expresia:

$$\zeta - (a_1 + a_2) \cdot \gamma + a_1 a_2 \cdot \xi = 0,$$

reprezintă ecuația unui plan în spațiul tridimensional $(\zeta\gamma\xi)$.

- cu metoda celor mai mici pătrate, se estimează parametrii planului de regresie care aproximează datele prelevate.

- pe baza rezultatelor de la punctul precedent, se estimează valorile constantelor de timp ale sistemului dinamic.

Se cere să se estimeze valorile numerice ale constantelor de timp ale modelului dinamic al dispozitivului pentru măsurarea modulului de elasticitate al pielii umane, T_{f1} și T_{f2}.

Problema 5.8

La portul de intrare al unui filtru se aplică un proces aleator zgomot alb pur cu densitatea spectrală de putere constantă $S_{ee} = 3$. Funcția de autocorelație a procesului aleator la portul de ieșire al filtrului este poate fi aproximată prin expresia:

$$R_{yy}(\tau) = 2.4 \cdot e^{-4|\tau|} \cos(2\tau).$$

Se cere să se deducă expresia funcției de transfer a filtrului.

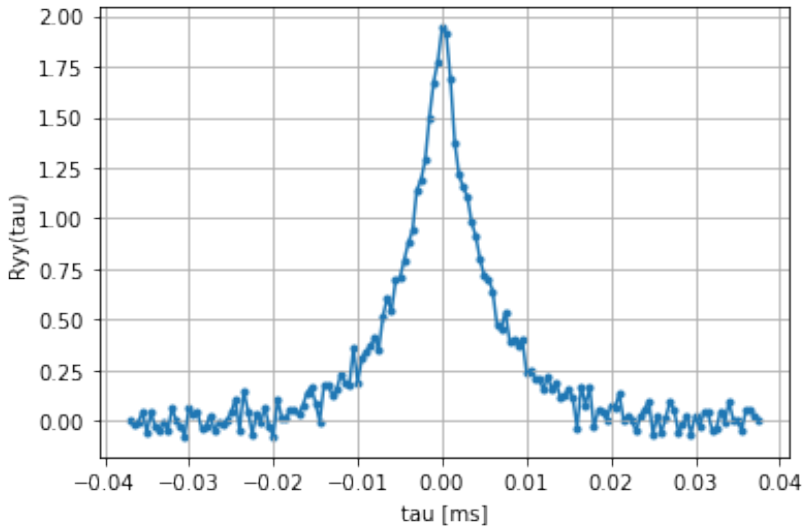


Figura 5.5: Eșantioanele funcției de autocorelație prelevate în experimentul din problema 5.9.

Problema 5.9

Într-un experiment de identificare a fost estimat modelul de zgomot al unei linii de măsurare. În acest acest scop, la portul de intrare al liniei a fost aplicat un proces aleator zgomot alb cu distribuție normală (gaussiană), media statistică zero și varianța unitară. A fost observat procesul aleator la portul de ieșire.
Pe baza datelor prelevate a fost estimată funcția de autocorelație a procesului aleator. Secvența de eșantioane este prezentată sub formă grafică în figura (5.5) și numerică în tabelele (5.8) și (5.9).

Pentru analiza datelor, experimentatorul a formulat ipoteza că procesul aleator este de tip zgomot colorat cu funcția de autocorelație reprezentat prin expresia generală:

$$R_{yy}(\tau) = \sigma^2 \cdot e^{-\mu|\tau|}.$$

Se cer următoarele.

(a) Pe baza datelor prelevate și a modelului adoptat, să se estimeze valorile parametrilor funcției de autocorelație a procesului.

(b) Să se calculeze expresia estimată a funcției densitate spectrală de putere $S_{yy}(\omega)$.

(c) Cu ipoteza suplimentară că modelul dinamic al liniei de măsurare este modelul unui sistem PT_1 să se estimeze valorile factorului de proporționalitate $\hat{a}$ și constantei de timp $\hat{b}$.

Tabela 5.5: Valorile eșantioanelor de timp [s] din problema 5.5.

0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104
108 112 116 120 124 128 132 136 140 144 148 152 156 160 164 168 172 176 180
184 188 192 196 200 204 208 212 216 220 224 228 232 236 240 244 248 252 256
260 264 268 272 276 280 284 288 292 296 300 304 308 312 316 320 324 328 332
336 340 344 348 352 356 360 364 368 372 376 380 384 388 392 396 400 404 408
412 416 420 424 428 432 436 440 444 448 452 456 460 464 468 472 476 480 484
488 492 496 500 504 508 512 516 520 524 528 532 536 540 544 548 552 556 560
564 568 572 576 580 584 588 592 596 600 604 608 612 616 620 624 628 632 636
640 644 648 652 656 660 664 668 672 676 680 684 688 692 696 700 704 708 712
716 720 724 728 732 736 740 744 748 752 756 760 764 768 772 776 780 784 788
792 796 800 804 808 812 816 820 824 828 832 836 840 844 848 852 856 860 864
868 872 876 880 884 888 892 896 900 904 908 912 916 920 924 928 932 936 940
944 948 952 956 960 964 968 972 976 980 984 988 992 996

Tabela 5.6: Valorile eșantioanelor vitezei unghiulare [$\frac{1}{min}$] din problema 5.5.

1630.7 1537.0 1542.0 1530.4 1484.9 1475.6 1489.6 1428.7 1404.6 1417.4 1346.4
1333.5 1287.3 1272.2 1285.5 1225.8 1210.9 1242.9 1165.5 1196.9 1168.3 1127.6
1108.2 1144.9 1078.8 1068.2 1073.5 1047.0 1050.9 994.3 977.9 934.3 995.8 908.5
965.0 921.3 875.9 893.3 892.6 859.8 852.9 824.2 850.6 802.0 799.6 757.6 782.6
773.9 761.8 759.2 709.7 718.5 676.4 658.5 637.3 658.8 650.9 644.2 644.2 608.0
594.9 609.4 594.8 603.4 554.3 531.8 540.4 526.3 551.1 563.9 553.4 491.5 534.5
499.8 466.5 496.0 466.2 432.7 490.4 422.5 425.4 471.9 468.2 445.2 426.8 386.0
377.4 367.2 417.5 387.2 391.5 357.8 370.9 344.7 335.7 342.1 307.6 354.9 312.2
340.0 311.9 346.3 323.5 315.1 341.9 263.0 270.0 279.7 304.8 283.0 306.2 241.9
264.4 289.2 230.9 247.5 274.9 241.1 240.3 273.9 203.9 220.7 257.2 184.4 230.8
233.2 180.9 221.9 239.1 230.5 162.7 224.8 156.8 201.1 226.6 171.3 196.5 150.0
166.0 182.0 170.7 158.5 186.0 155.3 123.7 131.7 172.8 173.1 166.0 156.1 141.9
106.1 147.2 121.1 123.3 151.3 105.5 118.7 119.6 142.0 157.3 107.2 97.9 79.6 104.9
118.2 98.3 146.4 75.5 98.9 106.4 75.3 128.7 101.5 131.2 96.6 113.6 81.0 83.4 81.0
73.7 61.2 102.3 59.8 114.7 68.6 115.6 62.3 45.5 110.1 75.1 66.0 44.4 61.9 52.0 45.0
100.2 95.1 43.0 54.3 34.9 90.0 66.7 81.9 46.3 20.8 49.8 68.2 80.7 55.1 82.7 24.9
68.2 30.7 45.5 57.2 61.9 35.1 26.0 10.7 56.7 15.7 59.9 50.0 52.5 50.0 30.7 67.3 30.8
59.3 14.1 41.9 62.7 45.2 37.3 47.6 25.0 62.8 57.5 20.7 13.3 8.5 60.9 -2.0 57.6 61.7
58.9 22.2 19.9 -5.5

Tabela 5.7: Valorile eșantioanelor de timp $[s]$ în experimentul din problema 5.7.

0.000	0.002	0.004	0.006	0.008	0.010	0.012	0.014	0.016	0.018	0.020	0.022	0.024
0.026	0.028	0.030	0.032	0.034	0.036	0.038	0.040	0.042	0.044	0.046	0.048	0.050
0.052	0.054	0.056	0.058	0.060	0.062	0.064	0.066	0.068	0.070	0.072	0.074	0.076
0.078	0.080	0.082	0.084	0.086	0.088	0.090	0.092	0.094	0.096	0.098	0.100	0.102
0.104	0.106	0.108	0.110	0.112	0.114	0.116	0.118	0.120	0.122	0.124	0.126	0.128
0.130	0.132	0.134	0.136	0.138	0.140	0.142	0.144	0.146	0.148	0.150	0.152	0.154
0.156	0.158	0.160	0.162	0.164	0.166	0.168	0.170	0.172	0.174	0.176	0.178	0.180
0.182	0.184	0.186	0.188	0.190	0.192	0.194	0.196	0.198	0.200	0.202	0.204	0.206
0.208	0.210	0.212	0.214	0.216	0.218	0.220	0.222	0.224	0.226	0.228	0.230	0.232
0.234	0.236	0.238	0.240	0.242	0.244	0.246	0.248	0.250	0.252	0.254	0.256	0.258
0.260	0.262	0.264	0.266	0.268	0.270	0.272	0.274	0.276	0.278	0.280	0.282	0.284
0.286	0.288	0.290	0.292	0.294	0.296	0.298						

Valorile eșantioanelor deplasării plunjerului în experimentul din problema 5.7.

0.0000007	0.0003401	0.0003991	0.0003906	0.0003884	0.0003568	0.0003686
0.0003530	0.0003516	0.0003497	0.0003430	0.0003424	0.0003367	0.0003316
0.0003120	0.0002921	0.0003111	0.0002934	0.0002676	0.0002780	0.0002635
0.0002530	0.0002663	0.0002706	0.0002525	0.0002455	0.0002458	0.0002267
0.0002248	0.0002395	0.0002263	0.0002015	0.0002129	0.0002205	0.0001821
0.0001925	0.0001967	0.0001816	0.0001893	0.0001768	0.0001873	0.0001670
0.0001716	0.0001613	0.0001743	0.0001471	0.0001674	0.0001660	0.0001456
0.0001439	0.0001711	0.0001418	0.0001480	0.0001434	0.0001024	0.0001540
0.0001318	0.0001235	0.0001258	0.0001237	0.0001121	0.0001063	0.0001137
0.0001243	0.0001045	0.0001160	0.0000957	0.0001084	0.0000997	0.0000851
0.0000805	0.0000825	0.0000996	0.0000795	0.0000779	0.0000956	0.0000799
0.0000943	0.0000702	0.0000625	0.0000643	0.0000753	0.0000627	0.0000776
0.0000668	0.0000741	0.0000637	0.0000504	0.0000445	0.0000495	0.0000843
0.0000619	0.0000569	0.0000613	0.0000610	0.0000731	0.0000500	0.0000604
0.0000394	0.0000412	0.0000306	0.0000474	0.0000309	0.0000393	0.0000414
0.0000370	0.0000454	0.0000360	0.0000346	0.0000375	0.0000395	0.0000458
0.0000333	0.0000174	0.0000256	0.0000362	0.0000148	0.0000451	0.0000162
0.0000326	0.0000372	0.0000242	0.0000140	0.0000253	0.0000215	0.0000101
0.0000212	0.0000148	0.0000278	0.0000091	0.0000163	0.0000437	0.0000360 -
0.0000061	0.0000211	0.0000496	0.0000128	0.0000036	0.0000151	0.0000135
0.0000153	0.0000188	0.0000206	0.0000113	0.0000181	0.0000033	0.0000009
0.0000109	0.0000034	0.0000145				

Tabela 5.8: Valorile eșantioanelor deplasării în domeniul timpului $[ms]$ în experimentul din problema 5.9.

-0.0370 -0.0365 -0.0360 -0.0355 -0.0350 -0.0345 -0.0340 -0.0335 -0.0330 -0.0325
-0.0320 -0.0315 -0.0310 -0.0305 -0.0300 -0.0295 -0.0290 -0.0285 -0.0280 -0.0275
-0.0270 -0.0265 -0.0260 -0.0255 -0.0250 -0.0245 -0.0240 -0.0235 -0.0230 -0.0225
-0.0220 -0.0215 -0.0210 -0.0205 -0.0200 -0.0195 -0.0190 -0.0185 -0.0180 -0.0175
-0.0170 -0.0165 -0.0160 -0.0155 -0.0150 -0.0145 -0.0140 -0.0135 -0.0130 -0.0125
-0.0120 -0.0115 -0.0110 -0.0105 -0.0100 -0.0095 -0.0090 -0.0085 -0.0080 -0.0075
-0.0070 -0.0065 -0.0060 -0.0055 -0.0050 -0.0045 -0.0040 -0.0035 -0.0030 -0.0025
-0.0020 -0.0015 -0.0010 -0.0005 0.0000 0.0005 0.0010 0.0015 0.0020 0.0025 0.0030
0.0035 0.0040 0.0045 0.0050 0.0055 0.0060 0.0065 0.0070 0.0075 0.0080 0.0085
0.0090 0.0095 0.0100 0.0105 0.0110 0.0115 0.0120 0.0125 0.0130 0.0135 0.0140
0.0145 0.0150 0.0155 0.0160 0.0165 0.0170 0.0175 0.0180 0.0185 0.0190 0.0195
0.0200 0.0205 0.0210 0.0215 0.0220 0.0225 0.0230 0.0235 0.0240 0.0245 0.0250
0.0255 0.0260 0.0265 0.0270 0.0275 0.0280 0.0285 0.0290 0.0295 0.0300 0.0305
0.0310 0.0315 0.0320 0.0325 0.0330 0.0335 0.0340 0.0345 0.0350 0.0355 0.0360
0.0365 0.0370 0.0375

Tabela 5.9: Valorile eșantioanelor funcției de autocorelație în experimentul din problema 5.9.

0.0018 -0.0152 -0.0042 0.0444 -0.0553 0.0401 -0.0278 -0.0437 -0.0068 -0.0488
0.0645 -0.0007 -0.0254 -0.0730 0.0639 0.0298 0.0435 -0.0384 -0.0283 0.0279 -0.0464
-0.0105 -0.0143 0.0046 0.0401 0.1097 -0.0471 0.1470 0.0494 -0.0641 0.0315 -0.0050
0.0643 -0.0264 -0.0771 0.1063 0.0181 0.0162 0.0566 0.0572 0.0298 0.0764 0.1396
0.1684 0.0891 -0.0032 0.1761 0.1774 0.1244 0.1529 0.2326 0.1857 0.1735 0.3597
0.1907 0.3118 0.3432 0.3685 0.4127 0.3553 0.5177 0.6095 0.5423 0.6945 0.7063
0.7930 0.8795 0.9402 1.1405 1.1880 1.2895 1.4992 1.6733 1.7754 1.9422 1.9129
1.6851 1.3775 1.2163 1.1530 1.1078 0.9806 0.9105 0.7967 0.7165 0.7018 0.6419
0.4753 0.4535 0.5398 0.3962 0.3975 0.3717 0.4068 0.2400 0.2444 0.2112 0.2106
0.1524 0.2216 0.1548 0.1898 0.1201 0.1260 0.1567 0.1133 -0.0326 0.1718 0.0791
0.1628 -0.0307 0.0521 0.0455 0.0065 0.0738 0.0649 0.1366 0.0063 0.0229 0.0008
-0.0509 0.0222 0.0589 0.0967 -0.0638 0.0265 -0.0565 0.0162 0.0968 0.0533 -0.0533
-0.0187 0.0275 -0.0643 0.0265 -0.0296 0.0397 0.0422 -0.0452 -0.0371 0.0494 -0.0054
0.0927 0.0081 0.0033 -0.0485 0.0502 0.0513 0.0252 -0.0004

Răspunsuri

Problema 5.1:

$\hat{a} = 0.208$; $\hat{b} = -0.022$

Problema 5.2:

(a) $y_l(t) = Y_0 e^{-\frac{t}{T_f}} \cdot \chi(t)$, parametrul care poate fi estimat este constanta de timp, T_f;

Problema 5.3:

(a) $\hat{a} = 0.009$; $\hat{b} = -0.349$; $\hat{c} = 3.702$; **(b)** $T_f = 8.136$ [min].

Problema 5.4:

(b) parametrul a reprezintă panta dreptei de regresie; **(c)** $y(t) = Y_0 \cdot e^{\frac{-\tau}{T_f}}$.

Problema 5.5:

$T_f = 250.286$ [s].

Problema 5.6:

(a) $g(t) = \frac{K_a e^{-\frac{t}{T_{f2}}}}{T_{f1}-T_{f2}} + \frac{K_a e^{-\frac{t}{T_{f1}}}}{T_{f1}-T_{f2}}$.

Problema 5.7:

$\hat{T}_{f1} = 0.006$, $\hat{T}_{f2} = 0.041$.

Problema 5.8:

$H(j\omega) = \frac{2.53(j\omega)+11.3}{(j\omega)^2+8(j\omega)+20}$.

Problema 5.9:

(a) $\sigma^2 = 1.85$, $\mu = 199.72$; **(b)** $S_{yy}(f) = \frac{0.018}{0.0001\pi^2 f^2+1.0}$; **(c)** $\hat{a} = 0.14$, $\hat{b} = 0.005$.

Rezolvări

Rezolvarea problemei 5.1

Etapele rezolvării.
Coeficienții dreptei de aproximare $\hat{a}$ și $\hat{b}$ se calculează cu metoda celor mai mici pătrate.
Se rezolvă următorul sistem de ecuații:

$$\begin{cases} \hat{a}\sum_{k=1}^{N} u_k^2 + \hat{b}\sum_{k=1}^{N} u_k &= \sum_{k=1}^{N} u_k \cdot y_k, \\ \hat{a}\sum_{k=1}^{N} u_k + \hat{b}\cdot N &= \sum_{k=1}^{N} y_k. \end{cases}$$

în care N este numărul de eșantioane ale secvențelor.
În continuare, pentru simplificare se notează $S1 = \sum_{k=1}^{N} u_k^2$, $S2 = \sum_{k=1}^{N} u_k$, $S3 = \sum_{k=1}^{N} u_k \cdot y_k$ și $S4 = \sum_{k=1}^{N} y_k$.

Coduri - Aplicația 5.1.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Se preiau datele prelevate.

```
aU=numpy.loadtxt('aU.txt')#preia datele din fisier
aYN=numpy.loadtxt('aYN.txt')#preia datele din fisier
aU,aYN
```

Se reprezintă graficul datelor prelevate.

```
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.plot(aU,aYN)
ax.scatter(aU,aYN)
ax.set_xlabel('Greutatea [kg]')
ax.set_ylabel('Elongatia [cm]')
ax.grid(True)
```

Rezultat: Graficul este reprezentat în figura(5.1).

Se calculează sumele $S1$, $S2$, $S3$ și $S4$.

```
S1=numpy.sum(aU*aU)#calculul sumei S1
S2=numpy.sum(aU)#calculul sumei S2
S3=numpy.sum(aU*aYN)#calculul sumei S3
S4=numpy.sum(aYN)#calculul sumei S4
N=numpy.size(aU)#numarul de esantioane
S1,S2,S3,S4,N
```

Rezultat: $617.5, 95.0, 126.2145, 19.299, 20$

Se rezolvă sistemul de ecuații.

```
a_est=sympy.symbols('a_est', real=True)#definitia variabilei a_est
b_est=sympy.symbols('b_est', real=True)#definitia variabilei b_est
expr=sympy.linsolve([S1*a_est+S2*b_est-S3,S2*a_est+b_est*N-S4],(
    a_est,b_est))#rezolvarea sistemului de ecuatii liniare
expr
```

```
a_est=list(expr)[0][0]#preia valoarea variabilei din structura de
    date
b_est=list(expr)[0][1]#preia valoarea variabilei din structura de
    date
a_est,b_est
```

```
format(a_est,'5.3f'),format(b_est,'5.3f')#afisare rezultate
```

Rezultat: $\hat{a} = 0.208$, $\hat{b} = -0.022$.

Se reprezintă grafic datele prelevate și aproximarea liniară a acestora.

```
aYest=a_est*aU+b_est
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.scatter(aU,aYN,label='Valori masurate')
ax.plot(aU,aYest, color='orange',label='Valori estimate')
ax.set_xlabel('Greutatea [kg]')
ax.set_ylabel('Elongatia [cm]')
ax.grid(True)
ax.legend()
```

Rezultat:

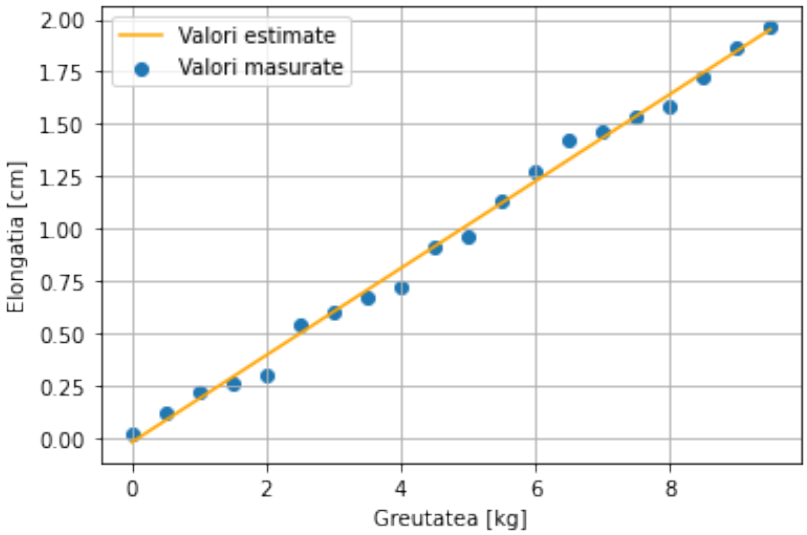


Figura 5.6: Reprezentarea grafică a datelor prelevate și aproximarea liniară a acestora.

Rezolvarea problemei 5.2

Etapele rezolvării.

(a) Calculul expresiei răspunsului liber cu condiție inițială diferită de zero.
Ecuația diferențială care descrie procesul este:

$$T_f \frac{dy(t)}{dt} + y(t) = 0,$$
$$y(0) = Y_0.$$

Se aplică transformarea Laplace și teorema derivării în domeniul timpului și se obține următoarea ecuație liniară în domeniul complex:

$$T_f \cdot [s \cdot Y(s) - y(0)] + Y(s) = 0.$$

Soluția ecuației de mai sus este transformata Laplace a răspunsului liber, $Y(s)$.
Pentru a obține expresia în domeniul timpului, se aplică transformarea Laplace inversă: $y(t) = \mathcal{L}^{-1}[Y(s)]$.

(b) Calculul diferenței momentelor pe axa timpului.
Expresia răspunsului liber calculată la punctul precedent fiind:

$$y(t) = Y_0 \cdot e^{\frac{-t}{T_f}},$$

valoarea argumentului t_1 care corespunde valorii $y_1 = y(t_1)$ a funcției este soluția ecuației:

$$Y_0 \cdot e^{\frac{-t_1}{T_f}} = y_1 \Rightarrow \ t_1 = T_f \log\left(\frac{Y_0}{y_1}\right).$$

Se procedează analog și pentru $y_2 = \frac{y_1}{e} = y(t_2)$ și se calculează diferența $\Delta_t = t_2 - t_1$.

(c) Abscisa punctului de intersecție a tangentei la grafic cu axa orizontală.
Ecuația dreptei tangente la graficul funcției $y(t)$ în punctul de cordonate $\left(0; \frac{Y_0}{Tf}\right)$ este:

$$(d) : y = y'(0_+) \cdot t + y(0) = -\frac{Y_0 \cdot t}{T_f} + Y_0,$$

rezultă că valoarea căutată a abscisei punctului t_A este soluția ecuației:

$$-\frac{Y_0 \cdot t_A}{T_f} + Y_0 = 0.$$

Coduri - Aplicația 5.2.
Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definitiile variabilelor și funcțiilor.

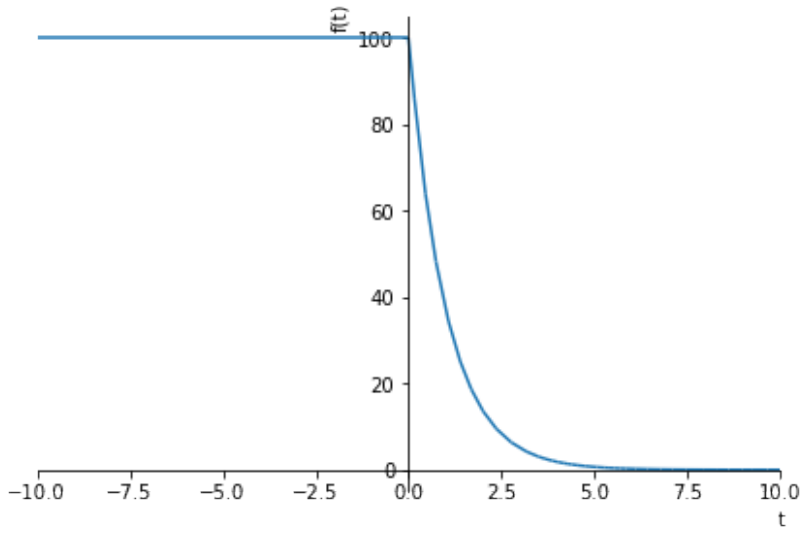


Figura 5.7: Graficul răspunsului liber cu condiție inițială nenulă al unui sistem $PT1$. Constanta de timp este $T_f = 1$.

```
s=symbols('s')#definitia variabilei complexe s
t=symbols('t', real=True)#defintia variabilei timp
y=Function('y')(t)#defintia raspunsului liber al sistemului dinamic
Y=Function('Y')#defintia transformatei Laplace a functiei y
Y_0=sympy.symbols('Y_0',real=True)#definitia constantei Y_0,
    valoarea functiei y inainte de momentul initial
```

Rezolvarea ecuației liniare care descrie procesul în domeniul complex.

```
Ys=sympy.Function('Ys')(s)#transformata Laplace a raspunsului liber
eq=sympy.solve(T_f*(s*Ys-Y_0)+Ys,Ys)[0]#calculeaza solutia ecuatiei
    in domeniul complex
eq
```

Rezultat: $Y(s) = \frac{T_f Y_0}{T_f s+1}$.

```
y=sympy.inverse_laplace_transform(eq,s,t)#calculeaza solutia
    ecuatiei in domeniul timnpului
y
```

Rezultat: $y(t) = Y_0 e^{-\frac{t}{T_f}} \theta(t)$.

Exemplu și reprezentare grafică.

```
dY0=100
dT_f=1
dY=y.subs(T_f,dT_f).subs(Y_0,dY0)+dY0*sympy.Heaviside(-t)
sympy.plot(dY,xlim=(-10,10)
```

Rezultat: în figura (5.7).

Calculul constantei de timp a procesului dinamic.

```
y1=sympy.symbols('y1',positive=True)#definitia variabilei y1 -
    valoarea functiei y la momentul t1
t1=sympy.solve(y-y1,t)[0]#calculul valorii t1 daca se cunoaste
    valoarea corespunzatoare a functiei y1
t1
```

Rezultat: $t_1 = T_f \log\left(\frac{Y_0}{y_1}\right)$,

```
y2=y1/exp(1)#valoarea functiei y care verifica relatia y2/y1=e
t2=sympy.solve(y-y2,t)[0]#calculul valorii t2 daca se cunoaste
    valoarea corespunzatoare a functiei y2
t2
```

Rezultat: $t_2 = T_f \log\left(\frac{eY_0}{y_1}\right)$,

```
delta_t=t2-t1#calculul expresiei diferentei dintre t2 si t1
delta_t
```

Rezultat: $-T_f \log\left(\frac{Y_0}{y_1}\right) + T_f \log\left(\frac{eY_0}{y_1}\right)$;

```
sympy.simplify(delta_t)#simplifica expresia de mai sus si afiseaza
    rezultatul
```

Rezultat: T_f.

Ecuația tangentei la graficul funcției $y(t)$ în punctul de abscisă $t = 0_+$.

```
y_plus=y.args[0]*y.args[2]#functia y se redefineste in intervalul
    (0;+oo)
y_tg=(sympy.diff(y_plus,t)).subs(t,0)*t+y_plus.subs(t,0)#ecuatia
    tangentei la graficul functiei y in origine
y_tg
```

Rezultat: $Y_0 - \frac{Y_0 t}{T_f}$,

```
sympy.solve(y_tg,t)[0]#calculeaza abscisa punctului de intersectie
    a tangentei cu axa orizontala si afiseaza rezultatul
```

Rezultat: T_f.

Rezolvarea problemei 5.3

Etapele rezolvării.

(a) Calculul valorilor coeficienților curbei de aproximare.
Coeficienții curbei de aproximare de ordinul doi, $\hat{a}$, $\hat{b}$ și $\hat{c}$ se calculează cu metoda celor mai mici pătrate și cu următorul sistem de ecuații:

$$\begin{cases} \hat{a}\sum\limits_{k=1}^{N} t_k^4 + \hat{b}\sum\limits_{k=1}^{N} t_k^3 + \hat{c}\sum\limits_{k=1}^{N} t_k^2 &= \sum\limits_{k=1}^{N} t_k^2 y_k, \\ \hat{a}\sum\limits_{k=1}^{N} t_k^3 + \hat{b}\sum\limits_{k=1}^{N} t_k^2 + \hat{c}\sum\limits_{k=1}^{N} t_k &= \sum\limits_{k=1}^{N} t_k y_k, \\ \hat{a}\sum\limits_{k=1}^{N} t_k^2 + \hat{b}\sum\limits_{k=1}^{N} t_k + \hat{c}\cdot N &= \sum\limits_{k=1}^{N} y_k. \end{cases}$$

În continuare, pentru simplificare se vor utiliza notațiile: $S1 = \sum\limits_{k=1}^{N} t_k^4$, $S2 = \sum\limits_{k=1}^{N} t_k^3$, $S3 = \sum\limits_{k=1}^{N} t_k^2$, $S4 = \sum\limits_{k=1}^{N} t_k$, $S5 = \sum\limits_{k=1}^{N} t_k^2 y_k$, $S6 = \sum\limits_{k=1}^{N} t_k y_k$ și $S7 = \sum\limits_{k=1}^{N} y_k$.

(b) Calculul constantei de timp a procesului.
Se procedează asemănător cu problema 5.2 punctul b:

- pe baza rezultatelor de la punctul (a), se construiește parabola care aproximează datele prelevate;
- valoarea y_1 se alege la 80% din valoarea maximă estimată;
- din ecuația parabolei de aproximare și cu valoarea y_1 se calculează valoarea momentului t_1;
- se procedează analog în cazul valorilor $y_2 = \frac{y_1}{e}$ și t_2;
- se calculează constanta de timp cu relația $T_f = t_2 - t_1$.

Coduri - Aplicația 5.3.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor prelevate din fisierele sursă.

```
aU=numpy.loadtxt('Prob53_aU.txt')#preia datele din fisier
aYN=numpy.loadtxt('Prob53_aYN.txt')#preia datele din fisier
aU,aYN
```

Reprezentarea grafică a datelor prelevate

```
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.plot(aU,aYN)
ax.scatter(aU,aYN)
```

```
ax.set_xlabel('Timpul [min]')
ax.set_ylabel('Inaltimea coloanei de apa [cm]')
ax.grid(True)
```

Rezultat: în figura (5.2)

Calculul sumelor $S1$, $S2$, $S3$, $S4$, $S5$, $S6$, $S7$.

```
S1=numpy.sum(numpy.power(aU,4))#calculul sumei S1
S2=numpy.sum(numpy.power(aU,3))#calculul sumei S2
S3=numpy.sum(numpy.power(aU,2))#calculul sumei S3
S4=numpy.sum(aU)#calculul sumei S4
S5=numpy.sum(numpy.power(aU,2)*aYN)#S5
S6=numpy.sum(aU*aYN)#S6
S7=numpy.sum(aYN)#S7
N=numpy.size(aU)#numarul de esantioane
S1,S2,S3,S4,S5,S6,S7,N
```

Rezultat:
$S1 = 3713541.6, S2 = 187578.1, S3 = 10106.2, S4 = 612.5, S5 = 4332.0, S6 = 376.3, S7 = 59.5, N = 50.$

Rezolvarea sistemului de ecuații.

```
a_est=sympy.symbols('a_est', real=True)#definitia variabilei a_est
b_est=sympy.symbols('b_est', real=True)#definitia variabilei b_est
c_est=sympy.symbols('c_est', real=True)#definitia variabilei c_est
expr=sympy.linsolve(
     [S1*a_est+S2*b_est+S3*c_est-S5,
      S2*a_est+S3*b_est+S4*c_est-S6,
      S3*a_est+S4*b_est+N*c_est-S7],
     (a_est,b_est,c_est))#rezolvarea sistemului de ecuatii liniare
expr
```

```
a_est=list(expr)[0][0]#preia valoarea variabilei din structura de
    date
b_est=list(expr)[0][1]#preia valoarea variabilei din structura de
    date
c_est=list(expr)[0][2]#preia valoarea variabilei din structura de
    date
a_est,b_est,c_est
```

Rezultat: $\hat{a} = 0.009, \hat{b} = -0.349,' \hat{c} = 3.702.$

Reprezentarea grafică a rezultatului.

```
aYest=a_est*aU**2+b_est*aU+c_est#valorile estimate
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.scatter(aU,aYN,label='Valori masurate')
ax.plot(aU,aYest, color='orange',label='Valori estimate')
ax.set_xlabel('Timpul [min]')
ax.set_ylabel('Inaltimea coloanei de apa [cm]')
ax.grid(True)
```

```
ax.legend()
```

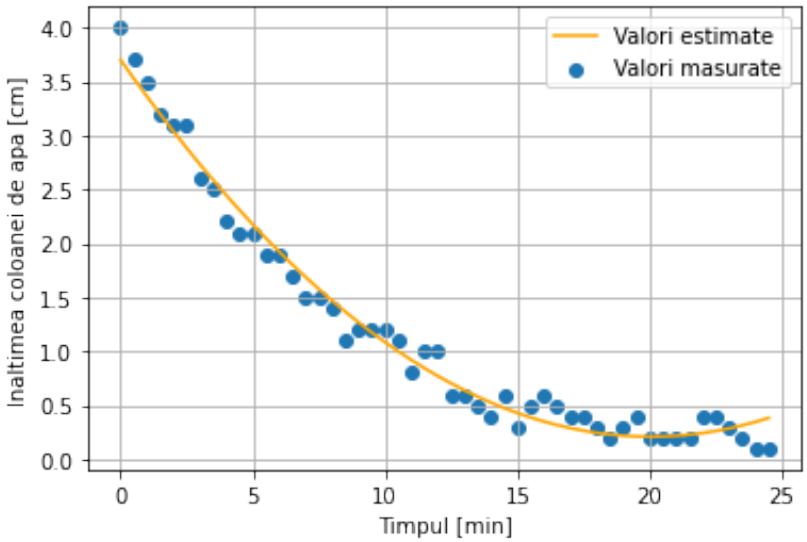


Figura 5.8: Variația înălțimii coloanei de apă în funcție de timp și aproximarea parabolică a acesteia; experimentul din problema 5.3.

Calculul constantei de timp.

```
t=sympy.symbols('t', real=True)#variabila simbolica t
y_est=a_est*t**2+b_est*t+c_est#expresia functiei polinomiale care
    aproximeaza setul de date
y_est
```

```
dY1=0.9*y_est.subs(t,0)#valoarea functiei y_1 a functiei
    polinomiale la momentul t_1 (se ia la 90\% din valoarea maxima)
dY2=dY1/sympy.exp(1).evalf()#valoarea y_2 a functiei polinomiale la
    momentul t_2
dT1=sympy.solve(y_est-dY1,t)[0]#calculul valorii momentului t_1
dT2=sympy.solve(y_est-dY2,t)[0]#calculul valorii momentului t_2
dTf=dT2-dT1#calculul constantei de timp
dTf# Nota: valoarea adevarata : 7.5 minute
```

Rezultat: $T_f = 8.136$ minute.

Rezolvarea problemei 5.4

Etapele rezolvării.

(a) Demonstrarea relației din enunț.

- Se pornește de la expresia funcției de transfer a procesului de tip $PT1$, $G(s) = \frac{K_a}{T_f \cdot s+1}$.

- Răspunsul pondere se calculează cu transformarea Laplace inversă:

$$g(t) = \mathcal{L}^{-1}G(s) = \mathcal{L}^{-1}\left[\frac{K_a}{T_f \cdot s + 1}\right] = \frac{K_a}{T_f} \cdot e^{\frac{-t}{T_f}}\chi(t).$$

- Se calculează expresia funcției $g(t+\tau)$ și apoi raportul celor două funcții $\frac{g(t+\tau)}{g(t)}$ în intervalul $(0; \infty)$:

$$\frac{g(t+\tau)}{g(t)} = \frac{\frac{K_a}{T_f} \cdot e^{\frac{-(t+\tau)}{T_f}}}{\frac{K_a}{T_f} \cdot e^{\frac{-t}{T_f}}} = e^{-\frac{\tau}{T_f}}.$$

(b) Semnificația factorului a.

Cu notațiile $x = g(t)$, $y = g(t+\tau)$ și $a = e^{-\frac{\tau}{T_f}}$ precum și pe baza rezultatului obținut la punctul precedent, se obține expresia:

$$y = a \cdot x;$$

care reprezintă o dreaptă în planul (xOy).
Factorul a este panta dreptei și dreapta trece prin originea sistemului de coordonate (ordonata la origine a dreptei este zero).

(c) Demonstrarea relației din enunț.

Expresiile răspunsului liber și răspunsului liber cu deplasare τ, în intervalul $(0, \infty)$ sunt date mai jos:

$$y(t) = Y_0 e^{-\frac{t}{T_f}}; \quad y(t+\tau) = Y_0 e^{-\frac{t+\tau}{T_f}}.$$

Apoi, se calculează raportul, $\frac{y(t+\tau)}{y(t)}$ și se procedează asemănător cu punctul precedent.

Coduri - Aplicația 5.4.
Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definitiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
tau=sympy.symbols('tau',positive=True)#definitia deplasarii in
    domeniul timpului
g=sympy.Function('g')(t)#defintia functiei pondere a sistemului
    dinamic
```

```
gTau=sympy.Function('gTau')(t)#defintia functiei pondere deplasata
    cu tau
y=sympy.Function('y')(t)#defintia raspunsului liber cu conditie
    initiala nenula
yTau=sympy.Function('yTau')(t)#defintia raspunsului liber cu
    conditie initiala nenula cu deplasarea tau
Gs=Function('Gs')#defintia transformatei Laplace a functiei pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
T_f=sympy.symbols('T_f',positive=True)#definitia constantei de timp
    T_f
Y0=sympy.symbols('Y0',real=True)#definitia valorii conditiei
    initiale nenule
```

Calculul expresiei funcției pondere.

```
Gs=K_a/(T_f*s+1)#expresia functiei de transfer
g=sympy.inverse_laplace_transform(Gs,s,t)#functia pondere se
    calculeaza cu transformarea Laplace inversa
Gs,g
```

Rezultat: $G(s)=\frac{K_a}{T_f s+1}$; $g(t)=\frac{K_a e^{-\frac{t}{T_f}}\chi(t)}{T_f}$.

Calculul funcției pondere deplasate cu τ și a raportului dintre cele două funcții.

```
gTau=g.subs(t,t+tau)#functia pondere deplasata
expr=sympy.simplify(gTau/g)
gTau,expr
```

Rezultat: $\frac{g(t+\tau)}{g(t)}=e^{-\frac{\tau}{T_f}}$.

Cazul răspunsului liber cu condiție inițială nenulă.

```
y=Y0*sympy.exp(-t/T_f)*sympy.Heaviside(t)
yTau=y.subs(t,t+tau)
expr1=sympy.simplify(yTau/y)
y,yTau,expr1
```

Rezultat: $y(t)=Y_0 e^{-\frac{t}{T_f}}\chi(t)$; $y(t+\tau)=Y_0 e^{-\frac{t+\tau}{T_f}}\chi(t+\tau)$; $\frac{y(t+\tau)}{y(t)}=e^{-\frac{\tau}{T_f}}$.

Rezolvarea problemei 5.5

ETAPELE REZOLVĂRII.
Se folosește metoda autofrânării.
Se alege o deplasare τ egala cu 10% din lungimea N a secvenței de date prelevate.
Apoi, cu secvența de date prelevate, $\{y_N\}_1^N$, se construiesc două secvențe de lungime $N-\tau$: o secvență $\{x\}_1^{N-\tau} = \{y_N\}_1^{N-\tau}$ și o a doua secvență $\{y\}_1^{N-\tau} = \{y_N\}_\tau^N$.
În continuare, se calculează parametrii dreptei de regresie în planul (xOy) asemănător cu problema 5.1.

CODURI - APLICAȚIA 5.5.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor prelevate din proces.

```
aU=numpy.loadtxt('Prob55_aTimp.txt')#preia datele din fisier
aYN=numpy.loadtxt('Prob55_aYN.txt')#preia datele din fisier
aU,aYN
```

Reprezentarea grafică a datelor prelevate.

```
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.plot(aU,aYN)
ax.set_xlabel('Timpul [s]')
ax.set_ylabel('Viteza unghiulara [1/min]')
ax.grid(True)
```

Rezultat: este prezentat în figura (5.3).
Preprocesarea datelor.

```
iN=aU.shape[0]#preia numarul de esantioane ale secventei
iTau=0.1*iN#indexul deplasarii = 10% din total
aX=aYN[0:int(iN)-int(iTau)]#vectorul aX
aY=aYN[int(iTau):int(iN)]#vactorul aY
```

Calculul dreptei de regresie.

```
S1=numpy.sum(aX*aX)#calculul sumei S1
S2=numpy.sum(aX)#calculul sumei S2
S3=numpy.sum(aX*aY)#calculul sumei S3
S4=numpy.sum(aY)#calculul sumei S4
N=numpy.size(aX)#numarul de esantioane
#S1,S2,S3,S4,N
a_est=sympy.symbols('a_est', real=True)#definitia variabilei a_est
b_est=sympy.symbols('b_est', real=True)#definitia variabilei b_est
expr=sympy.linsolve([S1*a_est+S2*b_est-S3,S2*a_est+b_est*N-S4],(
    a_est,b_est))#rezolvarea sistemului de ecuatii liniare
#expr
```

```
a_est=list(expr)[0][0]#preia valoarea variabilei din structura de
    date
b_est=list(expr)[0][1]#preia valoarea variabilei din structura de
    date
#a_est, b_est
aYest=a_est*aX+b_est
```

Graficul dreptei de regresie.

```
fig,ax = pyplot.subplots(nrows=1,ncols=1)
ax.scatter(aX,aY,marker='.')
ax.plot(aX,aYest,color='orange')
ax.set_xlabel('aX')
ax.set_ylabel('aY')
ax.grid(True)
```

Rezultat:

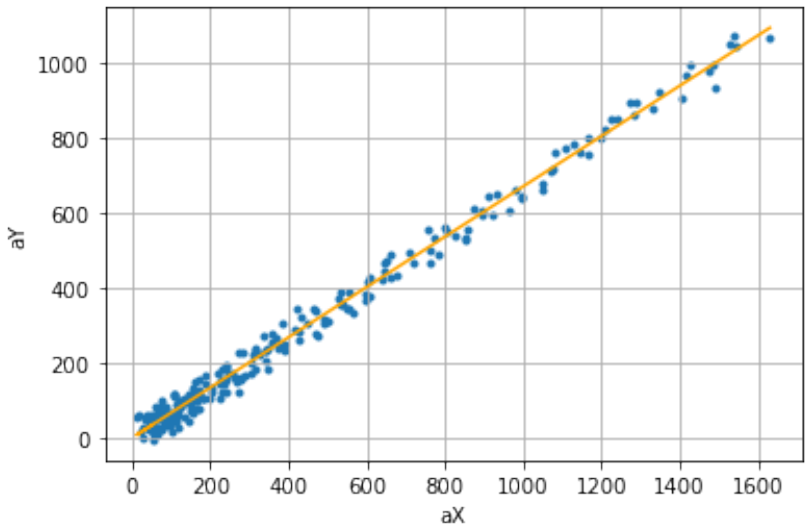


Figura 5.9: Dreapta de regresie pentru datele prelevate în experimentul din problema 5.

Calculul constantei de timp a procesului.

```
dTf=sympy.symbols('dTf',positive=True)
dTe=4#perioada de esantionare
dTau=iTau*dTe#deplasarea in timp
sympy.solve(a_est-exp(-dTau/dTf),dTf)[0]#valoarea adevarata 250
```

Rezultat: $T_f = 250.286$.

Rezolvarea problemei 5.6

ETAPELE REZOLVĂRII.
(a) Calculul expresiei funcției pondere.
Se utilizează transformarea Laplace inversă și relația:

$$g(t) = \mathcal{L}^{-1}[G(s)] \Rightarrow g(t) = \frac{K_a e^{-\frac{t}{T_{f2}}}}{T_{f1} - T_{f2}} + \frac{K_a e^{-\frac{t}{T_{f1}}}}{T_{f1} - T_{f2}}.$$

(b). Pe baza rezultatului de la punctul precedent, se calculează expresiile funcțiilor $g(t+\tau)$ și $g(t+2\cdot\tau)$ și apoi relația din enunț se verifică prin calcul direct.

(c). Relația de la punctul (b) se împarte prin $g(t)$, se introduc notațiile $\varphi = \frac{g(t+2\tau)}{g(t)}$ și $\xi = \frac{g(t+\tau)}{g(t)}$ și se rearanjează termenii, se obține expresia:

$$\varphi = (a_1 + a_2) \cdot \xi - a_1 \cdot a_2.$$

Expresia de mai sus reprezintă ecuația unei drepte în planul $(\xi O \varphi)$; panta dreptei este factorul $(a_1 + a_2)$ și ordonata la origine este termenul $-a_1 \cdot a_2$.

CODURI - APLICAȚIA 5.6.
Se apelează modulele necesare.

```
import sympy
from sympy import*
from matplotlib import pyplot
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
s=sympy.symbols('s')#definitia variabilei complexe s
t=sympy.symbols('t', real=True)#defintia variabilei timp
tau=sympy.symbols('tau',positive=True)#definitia deplasarii in
    domeniul timpului
g=sympy.Function('g')(t)#defintia functiei pondere a sistemului
    dinamic
gTau=sympy.Function('gTau')(t)#defintia functiei pondere deplasata
    cu tau
y=sympy.Function('y')(t)#defintia raspunsului liber cu conditie
    initiala nenula
yTau=sympy.Function('yTau')(t)#defintia raspunsului liber cu
    conditie initiala nenula cu deplasarea tau
Gs=Function('Gs')(s)#defintia transformatei Laplace a functiei
    pondere
K_a=sympy.symbols('K_a',real=True)#definitia factorului de
    proportionalitate K_a
T_f1=sympy.symbols('T_f1',positive=True)#definitia constantei de
    timp T_f1
T_f2=sympy.symbols('T_f2',positive=True)#definitia constantei de
    timp T_f2
```

Calculul expresiei funcției pondere.

```
Gs=K_a/((T_f1*T_f2*s**2+(T_f1+T_f2)*s+1))#expresia functiei de
    transfer
g=sympy.inverse_laplace_transform(Gs,s,t)#functia pondere se
    calculeaza cu transformarea Laplace inversa
Gs,g
```

Rezultat: $G(s)=\frac{K_a}{T_{f1}T_{f2}s^2+s\left(T_{f1}+T_{f2}\right)+1}$; $g(t)=-\frac{K_a e^{-\frac{t}{T_{f2}}}\xi(t)}{T_{f1}-T_{f2}}+\frac{K_a e^{-\frac{t}{T_{f1}}}\xi(t)}{T_{f1}-T_{f2}}$.

Calculul funcției pondere nedeplasate în intervalul $(0,\infty)$.

```
g0Tau=g.args[0].args[0]*g.args[0].args[1]*g.args[0].args[3]+g.args
    [1].args[1]*g.args[1].args[2]*g.args[1].args[4]
g0Tau
```

Rezultat: $\frac{K_a e^{-\frac{t}{T_{f2}}}}{T_{f1}-T_{f2}}+\frac{K_a e^{-\frac{t}{T_{f1}}}}{T_{f1}-T_{f2}}$

Calculul expresiilor funcției pondere deplasate cu τ și repsectiv cu 2τ.

```
g1Tau=g0Tau.subs(t,t+tau)
g2Tau=g0Tau.subs(t,t+2*tau)
g1Tau,g2Tau
```

Rezultat: $\frac{K_a e^{-\frac{t+\tau}{T_{f2}}}}{T_{f1}-T_{f2}}+\frac{K_a e^{-\frac{t+\tau}{T_{f1}}}}{T_{f1}-T_{f2}}$ și $\frac{K_a e^{-\frac{t+2\tau}{T_{f2}}}}{T_{f1}-T_{f2}}+\frac{K_a e^{-\frac{t+2\tau}{T_{f1}}}}{T_{f1}-T_{f2}}$.

Verificarea expresiei din enunțul problemei.

```
a1=sympy.symbols('a1',positive=True)
a2=sympy.symbols('a2',positive=True)
a1=sympy.exp(-tau/T_f1)
a2=sympy.exp(-tau/T_f2)
expr=g2Tau-(a1+a2)*g1Tau+a1*a2*g0Tau
sympy.simplify(expr)
```

Rezultat: 0.

Rezolvarea problemei 5.7

Etapele rezolvării.

- Se folosește formula estimatorului celor mai mici pătrate:

$$\hat{\theta} = \left(X^T \cdot X\right)^{-1} \cdot \left(X^T \cdot Y\right).$$

- Ecuația care descrie modelul este:

$$\zeta = \hat{a} \cdot \gamma + \hat{b} \cdot \xi + \hat{c}.$$

- Elementele matricelor X și Y din formula estimatorului se definesc pe baza ecuației modelului și a valorilor datelor prelevate ținând cont de relațiile $\zeta = g(t + 2\tau)$, $\gamma = g(t + \tau)$ și $\xi = g(t)$.

Coduri - Aplicația 5.7.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor prelevate.

```
aU=numpy.loadtxt('Prob57_aTimp.txt')#preia datele din fisier
aYN=numpy.loadtxt('Prob57_aYN.txt')#preia datele din fisier
aU.shape[0]
```

Reprezentarea grafică a datelor prelevate.

```
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.plot(aU,aYN)
ax.set_xlabel('Timpul [s]')
ax.set_ylabel('Viteza unghiulara [1/min]')
ax.grid(True)
```

Rezultat: în figura(5.4).

Preprocesarea datelor (1).

```
iN=aU.shape[0]#preia numarul de esantioane ale secventei
iTau=0.05*iN#indexul deplasarii = 5\% din total
aG=aYN[0:int(iN)-2*int(iTau)]#vectorul aG
aGtau=aYN[int(iTau):int(iN)-int(iTau)]#vectorul aGtau
aG2tau=aYN[2*int(iTau):int(iN)]#vectorul aG2tau
aG.shape,aGtau.shape,aG2tau.shape
```

```
fig = pyplot.figure()
ax = fig.add_subplot(projection='3d')
ax.scatter(aG,aGtau,aG2tau,marker='.')
```

```
ax.set_xlabel('aG')
ax.set_ylabel('aGtau')
ax.set_zlabel('aG2tau')
ax.grid(True)
```

Rezultat: în figura(5.10).

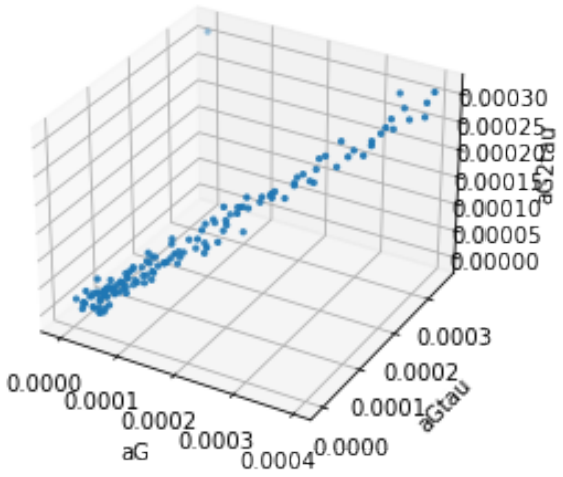


Figura 5.10: Reprezentarea în spațiu a punctelor de coordonate $(g(t), g(t+\tau), g(t+2\tau))$ în experimentul din problema 5.7.

Preprocesarea datelor (2).

```
aG0tau=numpy.reshape(aG,(aG.shape[0],1))#transformarea in vector
    coloana
aG1tau=numpy.reshape(aGtau,(aGtau.shape[0],1))#transformarea in
    vector coloana
aG2tau=numpy.reshape(aG2tau,(aG2tau.shape[0],1))#transformarea in
    vector coloana
aG0tau.shape,aG1tau.shape,aG2tau.shape
```

Inserarea valorilor elementelor in matricele X si Y

```
aY=numpy.empty([aG0tau.shape[0],1])#prototipul matricei aY
aX=numpy.empty([aG0tau.shape[0],3])#prototipul matricei aX
for k in list(range(aG0tau.shape[0])):
    aY[k,0]=aG2tau[k,0]
    aX[k,0]=aG1tau[k,0]
    aX[k,1]=aG0tau[k,0]
    aX[k,2]=1
aY.shape,aX.shape
```

Calculul parametrilor estimați cu estimatorul celor mai mici pătrate.

```
aTheta=numpy.matmul(numpy.linalg.inv(numpy.matmul(numpy.transpose(
    aX),aX)),(numpy.matmul(numpy.transpose(aX),aY)))
aTheta
```

Graficul planului de regresie.

```
fig = pyplot.figure()
ax = fig.add_subplot(projection='3d')
ax.scatter(aG,aGtau,aG2tau,marker='.',label='masurat')
ax.scatter(aG,aGtau,aY_est,color='orange',marker='.',label='estimat
    ')
ax.set_xlabel('aG')
ax.set_ylabel('aGtau')
ax.set_zlabel('aG2tau')
ax.grid(True)
ax.legend()
```

Rezultat: în figura (5.11)

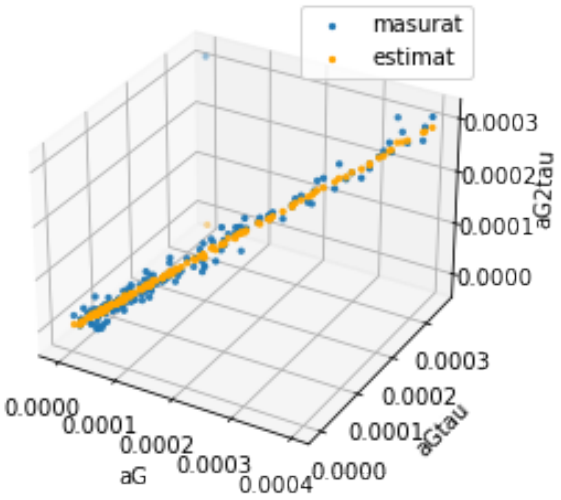


Figura 5.11: reprezentarea în spațiu a punctelor de coordonate $(g(t), g(t+\tau), g(t+2\tau))$ și a punctelor planului de aproximare în experimentul din problema 5.7.

Estimarea constantelor de timp ale procesului.

```
x=sympy.symbols('x',real=True)#definitia variabilei in ecuatia de
    gradul doi in S si P
S=sympy.symbols('S',real=True)#definitia sumei valorilor
    necunoscutelor
P=sympy.symbols('P',real=True)#definitia produsului valorilor
    necunoscutelor
S=aTheta[0][0]#preia valoarea sumei
P=aTheta[1][0]#preia valoarea produsului
eq=x**2-S*x+P#ecuatia de gradul doi in S si P
x1_est,x2_est=sympy.solve(eq,x)#solutiile ecuatiei de gradul doi
```

```
dTe=0.002#perioada de esantionare
dTau1=dTe*iTau#durata deplasarii in domeniul timpului
```

```
dTf1=sympy.symbols('dTf1',positive=True)#defintia constantei de
    timp Tf1
```

```
dTf2=sympy.symbols('dTf2',positive=True)#definitia constantei de
    timp Tf2
dT_f1=sympy.solve(sympy.exp(-dTau1/dTf1)-x1_est,dTf1)[0]#calculul
    valorii constantei de timp Tf1
dT_f2=sympy.solve(sympy.exp(-dTau1/dTf2)-x2_est,dTf2)[0]#calculul
    valorii constantei de timp Tf2
dT_f1,dT_f2
```

Rezultat: $\hat{T}_{f1} = 0.006$, $\hat{T}_{f2} = 0.041$.

Rezolvarea problemei 5.8

Etapele rezolvării.

- Fiind dată expresia funcției de autocorelație a procesului aleator la portul de ieșire $R_{yy}(\tau)$, se calculează expresia funcției densitate spectrală de putere $S_{yy}(\omega)$:

$$S_{yy}(\omega) = \mathcal{F}\left[R_{yy}(\tau)\right].$$

- Se calculează expresia estimată a $|H(j\omega)|^2$ cu formula:

$$S_{yy}(\omega) = |H(j\omega)|^2 \cdot S_{uu}(\omega),$$

în care $S_{uu}(\omega)$ se ia din datele problemei.

- Se adoptă prototipului funcției de frecvență a filtrului:

$$H(j\omega) = \frac{a(j\omega) + b}{c(j\omega)^2 + d(j\omega) + e}.$$

- Se calculează expresia $|H(j\omega)|^2$ pe baza prototipului funcției de frecvență.

- Parametrii a, b, c, d și e se obțin prin compararea coeficienților expresiei estimate a funcției de frecvență cu expresia calculată pe baza prototipului funcției.

Coduri - Aplicația 5.8.
Se apelează modulele necesare.

```
import numpy
import sympy
from sympy import*
sympy.init_printing()
```

Definițiile variabilelor și funcțiilor.

```
sigma=sympy.symbols('sigma',positive=True)#defintia variabilei
    sigma - varianta procesului aleator
zeta=sympy.symbols('zeta',positive=True)#definitia variabilei zeta
    - amortizarea procesului aleator
omega_r=sympy.symbols('omega_r',positive=True)#definitia variabilei
     omega_r
omega=sympy.symbols('omega',positive=True)#defintia parametrului
    omega_r - pulsatia de rezonanta
tau=sympy.symbols('tau',real=True)#definitia variabilei tau -
    deplasarea in domeniul timpului
f=sympy.symbols('f',real=True)#definitia variabilei frecventa
R=sympy.Function('R')(tau)#definitia functiei de autocorelatie
S=sympy.Function('S')(omega)#definitia functiei densitate spectrala
     de putere
```

Expresia funcției de autocorelație.

```
R=sigma*exp(-zeta*abs(tau))*cos(omega_r*tau).rewrite(exp)#expresia
    functiei de autocorelatie
R#afiseaza rezultatul
```

Rezultat: $\sigma\left(\frac{e^{i\omega_r\tau}}{2}+\frac{e^{-i\omega_r\tau}}{2}\right)e^{-\zeta|\tau|}$.

Calculul funcției densitate spectrală de putere.

```
S=sympy.fourier_transform(R,tau,f)#calculeaza densitatea spectrala
    de putere
simplify(S.subs(2*pi*f,omega))#afiseaza rezultatul
```

Rezultat: $\frac{2\sigma\zeta\left(\omega^2+\omega_r^2+\zeta^2\right)}{\omega^4-2\omega^2\omega_r^2+2\omega^2\zeta^2+\omega_r^4+2\omega_r^2\zeta^2+\zeta^4}$

Expresia funcției densitate spectrală de putere cu valori numerice.

```
dSigma=2.4#valoarea numerica a parametrului sigma
dZeta=4#valoarea numerica a parametrului zeta
dOmega_r=2#valoarea numerica a parametrului omega_r
R_num=R.subs(sigma,dSigma).subs(zeta,dZeta).subs(omega_r,dOmega_r)
R_num
```

Rezultat: $2.4\left(\frac{e^{2i\tau}}{2}+\frac{e^{-2i\tau}}{2}\right)e^{-4|\tau|}$.

```
S_num=S.subs(sigma,dSigma).subs(zeta,dZeta).subs(omega_r,dOmega_r)
S_num
```

Rezultat: $\frac{19.2\left(4\pi^2f^2+20\right)}{16\pi^4f^4+96\pi^2f^2+400}$.

Reformularea expresiei funcției densitate spectrală de putere ca funcție de pulsația ω.

```
S_num.args#se separa factorii care compun expresia
```

```
S_num_1=S_num.args[0]#primul factor ramane nemodificat
S_num_2=S_num.args[1].subs(2*sympy.pi*f,omega)#al doilea factor se
    substituie frecventa f cu pulsatia omega
S_num_3=S_num.args[2].subs(2*sympy.pi*f,omega)#al treilea factor se
    substituie frecventa f cu pulsatia omega
dSyy=S_num_1*S_num_2*S_num_3#se reface expresia functiei cu noua
    variabila
dSyy
```

Rezultat: $\frac{19.2\left(\omega^2+20\right)}{\omega^4+24\omega^2+400}$.

Calculul expresiei modulului funcției de frecvență la puterea 2.

```
dSuu=3#valoarea densitatii spectrale de putere a procesului aleator
    la portul de intrare
H2=dSyy/dSuu#expresia modulului la puterea 2
H2
```

Rezultat: $\frac{6.4\left(\omega^2+20\right)}{\omega^4+24\omega^2+400}$.

Estimarea parametrilor funcției de frecvență.

```
a=sympy.symbols('a', real=True)
b=sympy.symbols('b', real=True)
c=sympy.symbols('c', real=True)
d=sympy.symbols('d', real=True)
e=sympy.symbols('e', real=True)
Hest=(a*(sympy.I*omega)+b)/(c*(sympy.I*omega)**2+d*(sympy.I*omega)+
    e)#prototipul functiei de frecventa
Hest
```

Rezultat: $\frac{ia\omega+b}{-c\omega^2+id\omega+e}$.

```
H2est=(sympy.Abs(Hest))**2#modulul la puterea 2 al prototipului
    functiei de frecventa
H2est
```

Rezultat: $\frac{a^2\omega^2+b^2}{c^2\omega^4-2ce\omega^2+d^2\omega^2+e^2}$.

```
dA=sympy.solve(a**2-6.4,a)[1]#estimarea valorii parametrului a
dA
```

Rezultat: $\hat{a} = 2.53$.

```
dB=sympy.solve(b**2-6.4*20,b)[1]#estimarea valorii parametrului b
dB
```

Rezultat: $\hat{b} = 11.3$.

```
dC=sympy.solve(c**4-1,c)[1]#estimarea valorii parametrului c
dC
```

Rezultat: $\hat{c} = 1$.

```
dE=sympy.solve(e**2-400,e)[1]#estimarea valorii parametrului e
dE
```

Rezultat: $\hat{e} = 20$.

```
dD=sympy.solve(-2*dC*dE+d**2-24,d)[1]#estimarea valorii
    parametrului d
dD
```

Rezultat: $\hat{d} = 8$.

Rezolvarea problemei 5.9

Etapele rezolvării.

(a) Estimarea valorilor parametrilor funcției de autocorelație.

- Pentru estimarea valorii raportului $\frac{\sigma^2}{\mu}$ se folosește proprietatea:

$$\int_0^\infty R_{yy}(\tau)d\tau = \int_0^\infty \sigma^2 \cdot e^{-\mu\cdot\tau}d\tau = \frac{\sigma^2}{\mu}.$$

  Prin urmare se va calcula integrala numerică a secvenței valorilor funcției de autocorelație din datele problemei.

- Pentru estimarea parametrului μ se pornește de la observația că funcția de autocorelație satisface relația:

$$R_{yy}(\tau + \Delta_\tau) = R_{yy}(\tau) \cdot e^{-\mu\cdot\Delta_\tau}$$

  asemănător cu funcția pondere a sistemelor dinamice de tip $PT1$. În continuare, parametrul μ se estimează în același mod în care s-a procedat la problema 5.5.

(b) Expresia funcției densitate spectrală de putere.
Se calculează prin integrare cu formula $S_{yy}(\omega) = \mathcal{F}[R_{yy}(\tau)]$.

(c) Estimarea valorilor factorului de proporționalitate și constantei de timp.

- Se adoptă modelul de tip $PT1$ cu funcția de frecvență:

$$H(j\omega) = \frac{a}{b \cdot (j\omega) + 1}.$$

- Rezultă expresia modulului funcției de frecvență la puterea a doua:

$$|H(j\omega)|^2 = \frac{a^2}{b^2\omega^2 + 1}.$$

- Pe de altă parte, pe baza expresiilor funcțiilor densitate spectrală de putere rezultă:

$$S_{yy}(\omega) = |H(j\omega)|^2 \cdot S_{ee}(\omega) = |H(j\omega)|^2;$$

  în care s-a ținut cont că $S_{ee}(\omega) = 1$ deoarece varianța procesului aleator la portul de intrare este unitară și media statistică este nulă.

- Valorile estimate ale parametrilor funcției de frecvență se obțin prin identificare din expresia funcției $|H(j\omega)|^2$ rezultată pe baza modelului adoptat și respectiv expresia aceleiași funcții obținută pe baza funcțiilor densitate spectrală de putere la porturi.

Coduri - Aplicația 5.9.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Se preiau datele prelevate din experiment.

```
aU=numpy.loadtxt('Cap5Prob59_aTau.txt')#preia datele din fisier
aYN=numpy.loadtxt('Cap5Prob59_aRyyN.txt')#preia datele din fisier
#aU,aYN
```

Reprezentarea grafică a datelor.

```
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.plot(aU,aYN)
ax.set_xlabel('Tau_[s]')
ax.set_ylabel('Functia_de_autocorelatie')
ax.grid(True)
```

Rezultat: în figura(5.5).

Preprocesarea datelor - se preia doar secvența care corespunde valorilor pozitive ale variabilei τ.

```
iN=aU.shape[0]#preia numarul de esantioane ale secventei
aTau=numpy.empty([int(iN/2),1])#prototipul matricei aTau
aRyy=numpy.empty_like(aTau)#prototipul matricei aRyy
aU1=numpy.reshape(aU,[aU.shape[0],1])#transforma aU in vector
    coloana
aYN1=numpy.reshape(aYN,[aYN.shape[0],1])#transforma aU in vector
    coloana
```

```
aTau=aU1[int(iN/2):iN,0]
aRyy=aYN1[int(iN/2):iN,0]
```

Reprezentare grafică.

```
fig,ax = pyplot.subplots(nrows=1,ncols=1)
ax.scatter(aTau,aRyy,marker='.')
ax.plot(aTau,aRyy,color='orange')
ax.set_xlabel('aTau')
ax.set_ylabel('aRyy')
ax.grid(True)
```

Rezultat: în figura (5.12).

Calculul parametrului $\frac{\sigma^2}{\mu}$.

```
sigma2_mu=numpy.trapz(aRyy,aTau)#calculul integralei numerice cu
    metoda trapezelor
sigma2_mu
```

Rezultat: 0.009.

Calculul parametrului μ cu metoda analizei de corelație.

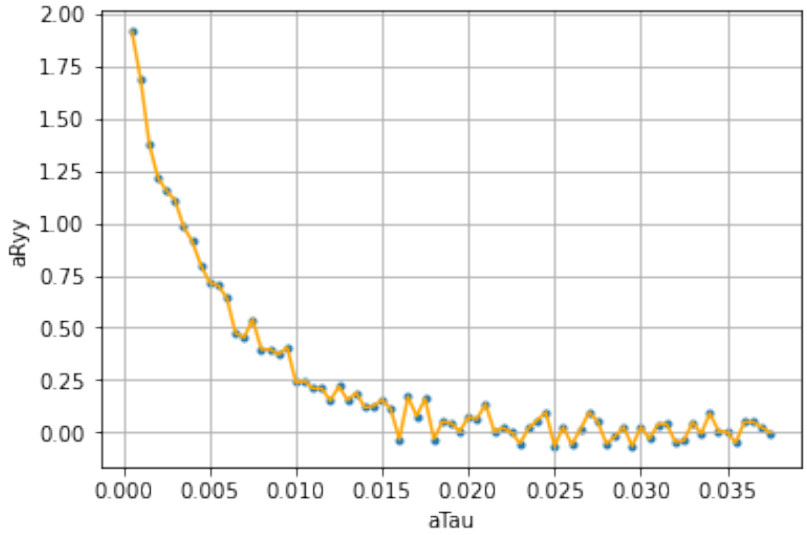


Figura 5.12: Graficul funcției de autocorelație în domeniul $\tau > 0$ pentru procesul aleator la portul de ieșire din experimentul prezentat în problema 5.9.

```
dDet=int(aTau.shape[0]*0.1)
aX=aRyy[0:(aRyy.shape[0]-dDet)]
aY=aRyy[dDet:aRyy.shape[0]]
aX.shape, aY.shape
```

```
S1=numpy.sum(aX*aX)#calculul sumei S1
S2=numpy.sum(aX)#calculul sumei S2
S3=numpy.sum(aX*aY)#calculul sumei S3
S4=numpy.sum(aY)#calculul sumei S4
N=numpy.size(aX)#numarul de esantioane
S1,S2,S3,S4,N
```

Rezultat: $S1 = 18.1852$, $S2 = 19.3659$, $S3 = 9.1521$, $S4 = 10.0219$, $N = 68$.

Calcul constantei de timp a procesului.

```
a_est=sympy.symbols('a_est', real=True)#definitia variabilei a_est
b_est=sympy.symbols('b_est', real=True)#definitia variabilei b_est
expr=sympy.linsolve([S1*a_est+S2*b_est-S3,S2*a_est+b_est*N-S4],(
    a_est,b_est))#rezolvarea sistemului de ecuatii liniare
expr
```

```
a_est=list(expr)[0][0]#preia valoarea variabilei din structura de
    date
b_est=list(expr)[0][1]#preia valoarea variabilei din structura de
    date
a_est,b_est
```

Rezultat: $\hat{a} = 0.4971$, $\hat{b} = 0.0058$.

```
aYest=a_est*aX+b_est
fig,ax=pyplot.subplots(nrows=1, ncols=1)
```

```
ax.scatter(aX,aY,label='Valori masurate')
ax.plot(aX,aYest, color='orange',label='Valori estimate')
ax.scatter(aX,aYest, color='orange',marker='o')
ax.set_xlabel('aX')
ax.set_ylabel('aY')
ax.grid(True)
ax.legend()
```

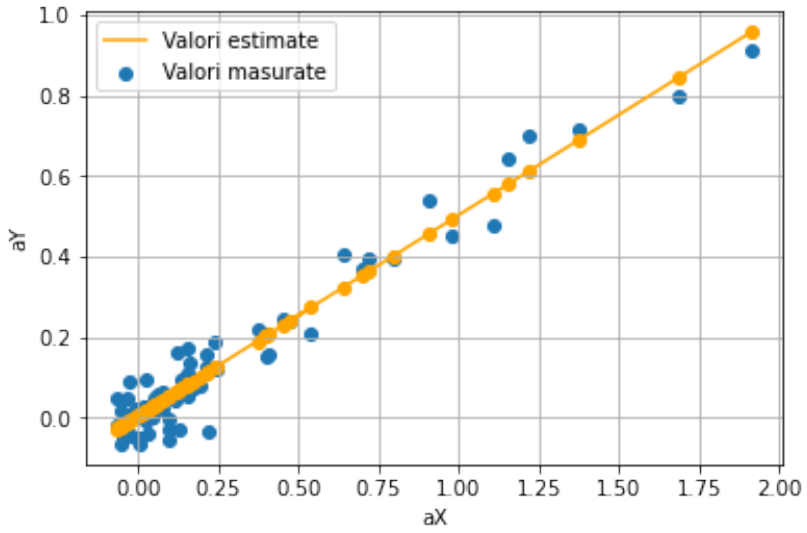


Figura 5.13: Dreapta de regresie și poziționarea regresorilor pentru datele din experimentul prezentat în problema 5.9.

```
dTe=0.0005#perioada de esantionare a semnalelor
mu=sympy.symbols('mu',positive=True)
dMu=sympy.solve(sympy.exp(-mu*dDet*dTe)-a_est,mu)[0]
dMu
```

Rezultat: $\mu = 199.72$.

```
dSigma2=sigma2_mu*dMu
dSigma2#valoarea adevarata 2
```

Rezultat: $\sigma^2 = 1.847$.

Funcția densitate spectrală de putere cu valori numerice.

```
tau=sympy.symbols('tau',real=True)
dRyy=dSigma2*sympy.exp(-dMu*sympy.Abs(tau))
dRyy
```

Rezultat: $R_{yy}(\tau) = 1.847e^{-199.72|\tau|}$.

```
f=sympy.symbols('f', real=True)# variabila f = frecventa
Syy=sympy.fourier_transform(dRyy,tau,f)
Syy
```

Rezultat: $S_{yy}(f) = \frac{0.018}{0.0001\pi^2 f^2 + 1.0}$.

Prototipul funcției de frecvență.

```
a=sympy.symbols('a',real=True)
b=sympy.symbols('b',real=True)
H=a/(b*sympy.I*(2*sympy.pi*f)+1)
H2=(sympy.Abs(H))**2
H2
```

Rezultat: $\frac{a^2}{4\pi^2 b^2 f^2 + 1}$.

```
a=sympy.sqrt(0.01849885)
b=sympy.sqrt(0.000100282839011909/4)
a,b#valorile adevarate a = 0.1 si b = 0.005
```

Rezultat: $\hat{a} = 0.14$, $\hat{b} = 0.005$.

6

Metode parametrice

Enunțuri

Problema 6.1

Pentru estimarea bias-ului (abaterii) unei linii de măsurare a procedează astfel: la capătul dinspre punctul de măsurare nu se aplică nici un semnal (semnalul este nul); la capătul opus se observă procesul aleator datorat exclusiv bias-ului și zgomotului pe linie. Cu ipoteza că zgomotul pe linie este un proces aleator cu media statistică zero, componenta continuă a procesului aleator observat reprezintă bias-ul liniei.
Calculele pentru estimarea valorii bias-ului se fac cu estimatorul celor mai mici pătrate și un model al procesului de forma:

$$y[k] = a + (e[k]);$$

în care $\{y\}_1^N$ este secvența eșatioanelor procesului observat, a este valoarea adevărată a bias-ului (parametrul modelului) și $\{e\}_1^N$ este secvența eșantioanelor zgomotului neobservabil.

Într-un experiment de identificare a bias-ului unei linii de măsurare a forțelor de frecare dintr-un stand de probă cu mase inerțiale, fost observată secvența de valori prezentată sub forma grafică în figura (6.1) și sub formă numerică în tabelul (6.1).

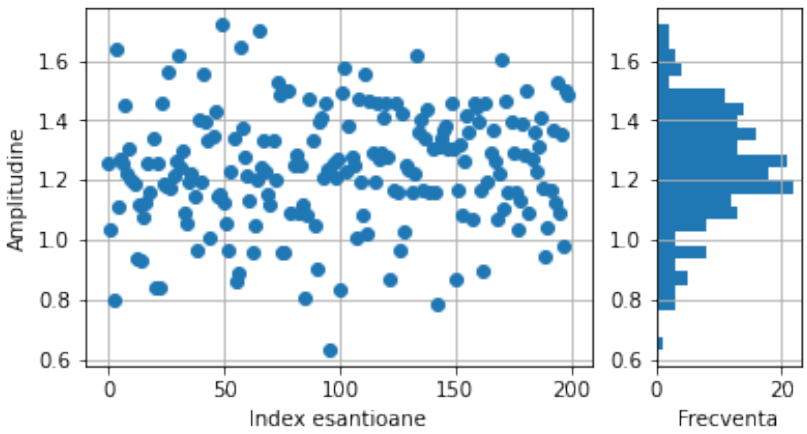


Figura 6.1: Secvența de valori observată în experimentul din problema 6.1.

Cu ajutorul procedurii prezentate în enunțul problemei și a datelor observate în experiment, se cere să se estimeze valoarea biasului liniei de măsurare.

Problema 6.2

Un inginer a dorit să verifice experimental concordanța dintre performanțele dinamice ale unui subansamblu dintr-o suspensie mecanică și valorile de catalog ale acestora. Pentru a realiza acest lucru, a aplicat la portul de intrare al sistemului dinamic testat un semnal proces aleator cu distributie normală, medie statistică zero și variantă unu și a observat procesul aleator la portul de ieșire. Zgomotul aditiv pe liniile de măsurare a fost neglijabil. Valorile secvențelor de eșantioane prelevate în cadrul experimentului sunt prezentate grafic în figura (6.2) și numeric în tabelele (6.2) și (6.3).
Se știe apriori că, sistemul dinamic examinat este definit printr-o funcție de transfer-z de forma generală:

$$H(z) = \frac{b}{z+a} = \frac{b \cdot z^{-1}}{1 + a \cdot z^{-1}}$$

și că valorile de catalog ale parametrilor modelului sunt $a = 0.5$ și $b = 0.7$.

Să se estimeze valorile $\hat{a}$ și $\hat{b}$ calculate pe baza modelului general și a secvențelor de eșantioane prelevate în cadrul experimentului. Ce se poate observa pe baza comparării rezultatelor?

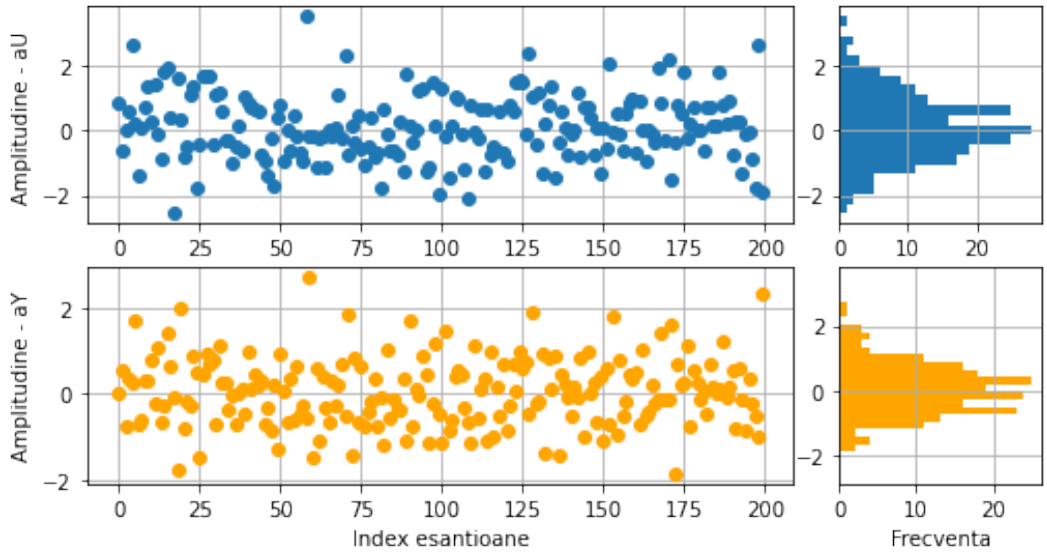


Figura 6.2: Secvențele de eșantioane prelevate la portul de intrare (graficele de sus) și la portul de ieșire (graficele de jos) în cadrul experimentului din enunțul problemei 6.2.

Problema 6.3

În experimentele de identificare a sistemelor - de cele mai multe ori - experimentatorul are la dispoziție doar secvențele de eșantioane ale proceselor aleatoare la porturile sistemului dinamic testat. Clasa de modele în care se încadrează modelul sistemului testat precum și caracteristicile statistice ale zgomotului perturbator pe linile de măsurare, nu se cunosc.

Inginerul care conduce experimentul trebuie să formuleze ipoteze menite să completeze informația care lipsește, apoi să implementeze procedura de identificare care furnizează valori pentru parametrii modelului și, la final să verifice dacă ipotezele pe baza cărora a efectuat experimentul sunt valide prin testarea rezultatului pe alte secvențe distincte de eșantioane.
La un examen, profesorul a cerut elevilor să identifice parametrii modelului unui sistem dinamic despre care se cunosc doar două perechi de secvențe de valori prelevate în aceleași condiții expreimentale la porturile intrare/ieșire.
Datele experimentale sunt prezentate sub formă grafică în figura (6.3) și sub formă numerică în tabelele (6.4) - (6.7).

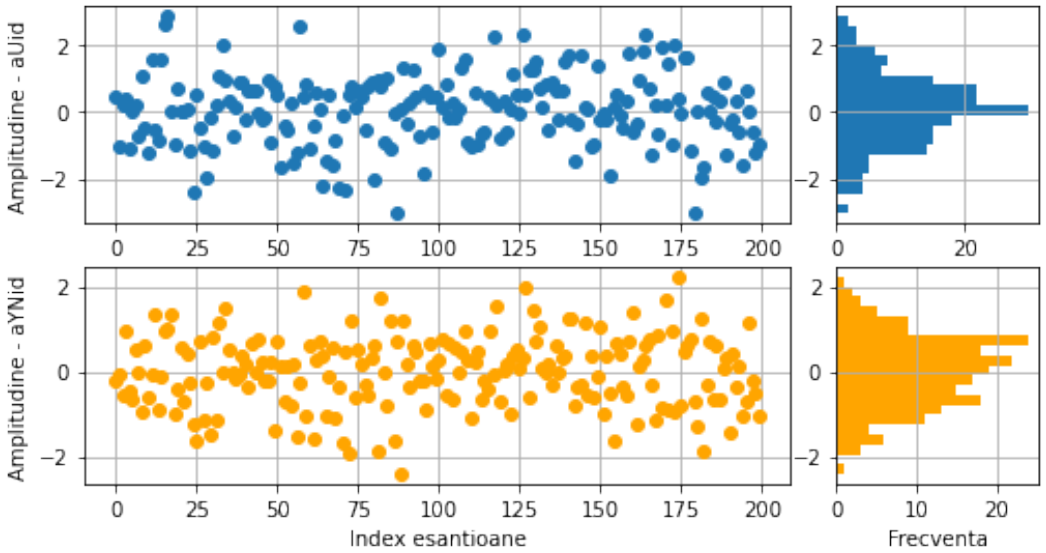


Figura 6.3: Secvențe de eșantioane prelevate la portul de intrare (graficele de sus) și la portul de ieșire (graficele de jos) în cadrul experimentului din enunțul problemei 6.3.

Problema 6.4

Trei grupe de studenți au efectuat un experiment de identificare pornind de la aceleași seturi de eșantioane la porturile de intrare/ieșire. Fiecare grupă a ales independent clasa de modele și algoritmul de identificare. Au rezultat trei modele candidate.

- Prima grupă de studenți a propus modelul reprezentat prin ecuația cu diferențe:

$$y[k] - 1.501y[k-1] + 0.55y[k-2] = u[k-1] + 0.3u[k-2] + (e[k]).$$

- A doua grupă de studenți a propus modelul reprezentat prin ecuația cu diferențe:

$$y[k] - 1.491y[k-1] + 0.707y[k-2] = u[k-1] + 0.551u[k-2] + (e[k]).$$

- A treia grupă de studenți a propus modelul reprezentat prin ecuația cu diferențe:

$$\begin{aligned} y[k] - 1.501y[k-1] + 0.55y[k-2] + 0.152y[k-3] = \\ u[k-1] + 0.435u[k-2] + 0.25u[k-3] + (e[k]). \end{aligned}$$

$e[k]; k = 1 \dots N$ reprezintă eșantionul la pasul k al zgomotului nobservabil pe liniile de măsurare.

Care dintre cele trei modele candidate aproximează cel mai bine modelul adevărat al sistemului dinamic? Pentru a-i ajuta pe studenți să găsească răspunsul la această întrebare, profesorul le-a pus la dispoziție încă două seturi de eșantioane la porturile de intrare/ieșire - disponibile în tabelele (6.8) și (6.9) - și recomandarea să compare valorile funcțiilor criteriu calculate pentru fiecare din cele trei modele candidate.

Problema 6.5

Diferența dintre valoarea măsurată și valoarea estimată pe baza modelului rezultat din identificare se numește eroare de aproximare sau reziduu. Analiza erorilor de aproximare oferă informații importante pentru validarea rezultatelor identificării. Compararea histogramelor (densitățile de probabilitate empirice) erorilor de aproximare calculate pentru două sau mai multe modele candidate permite selectarea modelului care aproximează cel mai bine modelul adevărat (necunoscut) al sistemului dinamic examinat dintr-un set de modele candidate - metoda verosimilității maxime.

Se cere să se aplice metoda verosimilității maxime pentru cazul celor trei modele candidate din problema 6.4 și pe această bază, să se selecteze modelul care aproximează cel mai bine modelul adevărat - dar care nu poate fi cunoscut - al sistemului dinamic examinat.

Problema 6.6

Un inginer și-a propus să analizeze regimul dinamic al unei instalații industriale. Pentru acest scop a folosit legi care descriu fenomene și proprietăți de material, specifice domeniului procesului analizat. Pe baza acestei abordări, inginerul a propus următoarea reprezentare a funcției de transfer-z a instalației:

$$H(z) = \frac{z + b_1}{z^2 + a_1 \cdot z + a_2};$$

în care; $a_1 = 0.74$, $a_2 = 0.02$ și $b_1 = 0.14$.
În continuare, inginerul a efectuat un experiment de identificare a sistemelor: a aplicat un proces aleator zgomot alb discret la portul de intrare (media statistică zero și varianța unu) și a observat procesul aleator la portul de ieșire.
Datele prelevate experimental sunt prezentate sub formă numerică în tabelele, (6.10) și (6.11).
Cu ajutorul datelor prelevate, se cere să se verifice dacă modelul propus de inginer reprezintă cea mai bună aproximare a modelului adevărat al procesului dinamic.

Problema 6.7

Intr-un experiment de identificare a unui sistem dinamic, au fost analizate erorile de aproximare - numite și reziduurile estimării.
În acest sens, a fost prelevată o secvență de 50 de eșantioane ale erorilor de aproximare și a fost evaluată corelația statistică dintre eșantioane. A fost ales un decalaj de $\tau = 5$ eșantioane și au fost create două secvențe cu lungimea de 45 eșantioane. Analiza a

constat în evaluarea valorii pantei dreptei de regresie dintre valorile celor două secvențe. Valorile eșantioanelor celor două secvențe sunt prezentate în Tabela 6.12 și respectiv în Tabela 6.13. In figura 6.4 este reprezentată dependența dintre eșantioanele celor două secvențe. Pentru estimarea parametrilor dreptei de regresie a fost implementată metoda celor mai mici pătrate.

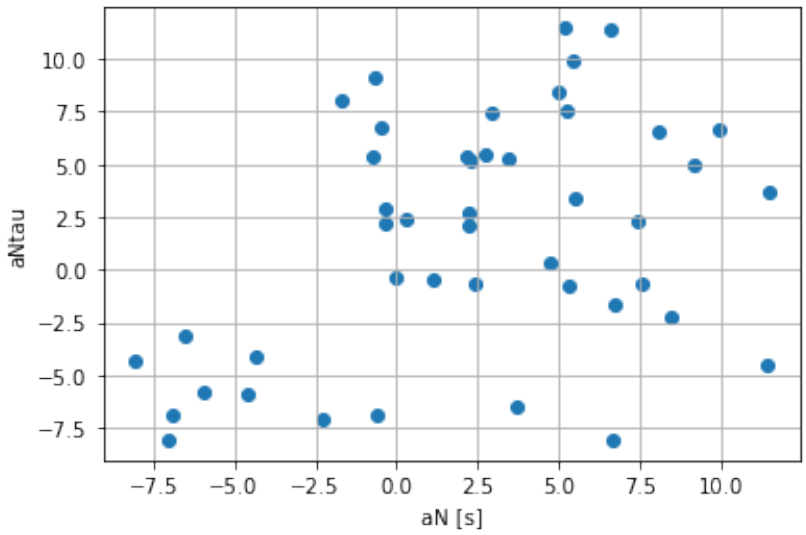


Figura 6.4: Dependența dintre eșantioanele secvenței erorilor de aproximare și secvența erorilor decalată cu τ din enunțul problemei 6.7.

Se cere să se estimeze:

(a) valorile pantei dreptei de regresie a_{est} și ordonatei la origine b_{est},

(b) unghiul în grade α^0 al dreptei de regresie în raport cu axa orizontală și,

(c) să se evalueze dacă erorile de aproximare analizate sunt de natura zgomotului alb.

Tabela 6.1: Secvența de valori observată în experimentul pentru estimarea bias-ului liniei de măsurare din problema 6.1.

0.895	1.728	1.566	1.188	1.194	1.070	1.403	1.396	1.269	1.060	1.198	1.111	1.412
1.264	1.487	1.449	1.360	1.383	1.496	1.227	1.195	1.013	1.318	1.295	1.110	1.226
1.348	1.115	1.809	1.143	1.073	1.400	1.249	1.433	1.084	1.215	1.371	0.670	1.287
1.385	1.412	1.200	1.111	0.936	1.348	1.507	1.247	1.530	1.379	1.184	1.273	1.303
1.196	1.478	1.152	1.153	0.984	1.025	1.151	1.135	1.115	1.361	1.179	1.501	1.319
1.107	1.064	1.708	1.485	1.591	1.217	1.456	1.388	1.322	1.438	1.356	1.161	1.134
1.690	1.231	1.251	1.169	1.402	1.143	1.410	1.276	1.335	1.207	1.527	1.197	0.951
1.392	1.110	1.515	1.314	1.093	1.401	1.322	1.541	1.219	0.897	1.572	1.612	1.240
1.440	1.516	1.225	1.177	1.537	1.461	0.992	0.922	1.243	1.401	1.049	1.071	1.438
1.513	1.362	1.260	0.960	1.318	1.316	1.147	1.199	0.990	1.207	0.832	1.152	0.966
1.190	1.320	1.181	1.455	1.461	1.311	1.433	1.245	1.681	1.270	0.888	1.182	0.908
1.428	1.283	0.830	1.283	1.106	1.002	0.951	1.130	1.090	1.221	1.460	1.531	1.100
1.259	1.226	1.225	1.544	1.355	1.600	1.588	1.151	1.171	1.589	1.223	1.416	1.395
1.337	1.284	0.812	1.166	1.044	1.015	1.031	1.502	1.296	0.921	1.441	1.485	0.936
1.272	1.638	1.286	1.295	1.278	1.200	1.033	1.150	1.643	1.105	1.197	1.371	1.495
1.595	1.353	1.467	1.103	1.343								

Tabela 6.2: Secvența de valori la portul de intrare observată în experimentul pentru verificarea concordanței parametrilor cu datele de catalog din problema 6.2.

0.819 -0.645 0.010 0.621 2.626 0.189 -1.353 0.057 0.704 1.358 0.265 1.415 -0.121 -0.869 1.788 1.911 0.375 -2.554 1.630 0.332 -0.790 -0.472 1.080 1.335 -1.741 -0.428 1.677 1.655 1.648 -0.405 1.118 1.182 0.589 -0.312 -0.296 -1.031 -0.472 0.178 -0.589 1.052 0.874 0.725 0.642 0.603 -0.771 -0.962 -1.411 -0.263 -1.683 0.421 0.769 -0.920 0.045 -0.599 0.462 -0.194 -0.713 -0.964 3.516 -0.183 -0.202 -1.111 -0.212 -0.189 -1.137 0.019 -0.151 0.086 1.129 -0.225 2.293 -0.721 0.159 -0.332 0.466 -0.577 -1.081 -0.503 0.383 -0.827 -0.655 -1.734 0.637 -0.088 -0.611 -0.645 -0.763 0.294 -1.253 1.714 0.177 -0.335 0.026 1.220 1.272 -1.281 -1.102 1.503 0.145 -1.960 1.284 -0.151 -1.449 0.150 1.052 1.004 -0.136 -1.169 -2.066 0.784 -0.013 -0.224 0.631 -1.275 0.638 -0.737 -0.461 -0.628 0.606 -0.698 -0.952 0.784 0.613 1.477 1.530 1.482 -0.109 2.381 1.034 -0.412 1.192 -1.314 0.190 0.765 1.356 -1.427 -0.382 0.608 0.068 -0.758 -0.112 0.043 1.186 -0.796 0.719 0.708 0.384 -0.743 0.092 -1.296 0.085 -0.553 2.074 -0.049 0.499 -0.136 -0.598 0.554 0.825 0.953 -0.666 0.941 -0.005 -0.928 0.026 -0.131 -0.337 1.926 0.831 -0.277 2.178 -1.524 -0.363 0.793 0.545 1.804 -0.254 0.240 0.747 0.025 -0.009 -0.597 0.690 0.733 0.118 0.164 1.834 0.758 0.205 0.897 -0.760 0.282 0.255 -1.289 -0.098 -0.048 -0.873 -1.750 2.626 -1.915

Tabela 6.3: Secvența de valori la portul de ieșire observată în experimentul pentru verificarea concordanței parametrilor cu datele de catalog din problema 6.2.

0.000 0.573 -0.738 0.376 0.246 1.715 -0.725 -0.585 0.332 0.326 0.787 -0.208 1.095 -0.632 -0.292 1.398 0.639 -0.057 -1.759 2.021 -0.778 -0.164 -0.248 0.881 0.494 -1.466 0.434 0.957 0.680 0.813 -0.690 1.128 0.264 0.280 -0.358 -0.028 -0.708 0.024 0.112 -0.469 0.971 0.126 0.445 0.227 0.308 -0.694 -0.326 -0.824 0.228 -1.292 0.941 0.068 -0.678 0.370 -0.605 0.626 -0.449 -0.275 -0.538 2.730 -1.493 0.605 -1.080 0.392 -0.328 -0.632 0.329 -0.270 0.196 0.693 -0.504 1.857 -1.433 0.828 -0.646 0.649 -0.729 -0.392 -0.156 0.346 -0.752 -0.082 -1.172 1.032 -0.578 -0.139 -0.382 -0.343 0.377 -1.066 1.733 -0.743 0.136 -0.050 0.879 0.451 -1.122 -0.210 1.157 -0.477 -1.134 1.465 -0.838 -0.595 0.402 0.535 0.435 -0.313 -0.662 -1.115 1.107 -0.562 0.124 0.380 -1.082 0.988 -1.010 0.183 -0.531 0.690 -0.834 -0.249 0.673 0.093 0.987 0.578 0.749 -0.451 1.892 -0.222 -0.178 0.923 -1.382 0.823 0.124 0.888 -1.442 0.454 0.199 -0.052 -0.505 0.174 -0.057 0.859 -0.987 0.997 -0.003 0.270 -0.655 0.392 -1.103 0.611 -0.692 1.798 -0.934 0.816 -0.503 -0.167 0.471 0.342 0.496 -0.714 1.016 -0.512 -0.393 0.215 -0.199 -0.137 1.416 -0.126 -0.131 1.590 -1.862 0.677 0.216 0.273 1.127 -0.741 0.538 0.254 -0.110 0.049 -0.442 0.704 0.161 0.002 0.114 1.227 -0.082 0.185 0.536 -0.800 0.597 -0.120 -0.842 0.353 -0.210 -0.506 -0.972 2.324

Tabela 6.4: Secvența de valori la portul de intrare folosită pentru identificarea parametrilor în experimentul din problema 6.3.

0.439 -1.029 0.198 0.383 -1.068 0.039 0.179 -0.717 1.043 -0.457 -1.209 1.531 -0.540 -0.827 1.534 2.617 2.843 0.042 -0.941 0.701 0.035 0.008 0.068 -1.148 -2.384 0.479 -0.488 -1.005 -1.939 -0.157 -1.135 0.173 1.075 1.990 0.967 0.346 -0.746 0.126 0.851 0.861 -0.216 0.602 0.654 0.646 -0.024 -0.233 -0.157 0.958 -0.905 0.766 0.488 -1.669 -0.434 -0.545 0.284 -1.555 -1.211 2.543 0.419 0.804 -1.077 -0.432 0.584 0.086 -2.209 -1.495 0.493 -1.615 -0.854 -2.236 -0.097 -2.310 0.496 0.755 0.160 0.612 0.458 -0.548 0.767 0.857 -2.027 0.948 0.750 -0.921 0.996 -1.102 -0.071 -3.029 0.062 1.335 -0.369 0.273 1.273 0.457 -0.708 -1.824 0.622 0.421 -0.610 0.422 1.887 0.111 0.789 -0.181 0.236 -0.154 0.063 1.300 1.531 -0.939 -1.014 0.590 -0.973 -0.687 -0.603 0.452 0.368 2.211 0.194 -0.798 0.347 -0.623 0.074 1.117 0.539 -0.028 2.313 0.497 1.273 1.226 1.509 0.678 0.169 0.803 -0.520 0.857 0.604 -0.233 0.615 1.473 1.690 -0.202 -1.471 -0.336 1.683 0.126 0.049 -1.055 -0.950 1.371 -0.226 -0.038 -0.245 -1.910 0.143 0.495 -0.121 -0.448 0.310 1.715 -0.586 -0.369 0.952 1.792 2.282 0.719 -1.282 0.181 -0.691 1.907 0.179 1.432 -0.959 2.011 0.408 -0.055 1.593 1.603 -1.157 -2.989 0.011 -1.946 -1.640 0.551 0.018 0.432 -0.309 0.294 0.858 -1.278 -0.353 -0.213 0.327 -0.596 -1.572 0.634 0.005 -0.583 -1.221 -0.989

Tabela 6.5: Secvența de valori la portul de ieșire folosită pentru identificarea parametrilor în experimentul din problema 6.3.

-0.217 -0.057 -0.572 0.945 -0.442 -0.658 0.524 -0.023 -0.957 0.608 -0.621 -0.092 1.328 -0.903 -0.128 0.958 0.978 1.364 -1.008 -0.434 0.541 -0.729 0.421 -0.291 -1.247 -1.651 0.726 -1.144 -0.284 -1.473 0.819 -1.125 1.124 -0.019 1.501 0.491 -0.000 -0.176 -0.053 0.387 0.152 -0.364 0.642 -0.003 0.761 0.197 -0.218 -0.210 0.243 -1.377 0.694 0.106 -0.704 0.127 -0.790 0.191 -1.540 -0.287 1.868 -1.048 0.609 -1.564 0.286 0.703 0.385 -1.055 -0.139 0.566 -1.118 -0.350 -1.675 0.472 -1.932 1.193 -0.588 0.504 0.159 -0.332 -0.555 0.310 0.625 -1.859 1.736 -0.039 -0.792 1.204 -1.627 0.727 -2.416 1.199 0.189 -0.366 0.616 0.488 -0.239 -0.238 -0.913 0.642 0.130 -0.149 0.276 0.772 -0.581 0.654 -0.675 0.551 -0.005 0.484 0.977 0.249 -1.112 0.215 0.478 -0.665 -0.207 -0.430 0.935 -0.069 1.539 -0.692 0.017 0.169 -1.009 0.369 0.495 0.088 0.312 1.966 -0.615 1.424 0.711 1.054 0.064 -0.013 0.148 -0.306 0.609 0.001 0.302 0.354 1.270 1.232 -0.798 -0.384 -0.338 1.156 -0.576 0.389 -0.592 -0.143 1.051 -0.988 0.362 -0.531 -1.652 0.656 0.404 -0.356 -0.576 0.696 1.411 -1.228 0.144 0.291 0.620 0.782 -0.313 -1.143 0.862 -0.829 1.690 -0.922 0.959 -0.933 2.230 -0.827 0.473 0.630 0.734 -0.730 -1.298 1.250 -1.893 -0.335 0.703 -0.637 0.634 -0.671 0.083 0.299 -1.456 0.392 -0.349 0.141 -1.064 -0.694 1.139 -0.206 -0.532 -1.033

Tabela 6.6: Secvența de valori la portul de intrare folosită pentru validare în experimentul din problema 6.3.

0.394 -1.110 -0.410 -0.507 -0.151 0.531 1.896 -1.271 2.576 0.586 -1.726 0.140 -0.643 0.686 -1.191 -2.275 1.315 -0.532 -0.665 0.039 -0.064 -0.740 0.053 1.161 1.567 -0.209 -2.265 1.090 0.859 -0.483 -0.225 1.024 -0.297 -0.963 2.296 -1.600 0.254 0.251 0.063 0.179 -1.819 0.490 1.288 -1.202 0.575 -0.589 -0.402 -0.382 -1.059 0.857 -0.502 -1.024 0.351 0.716 -0.685 -1.157 2.065 0.953 -2.082 -2.487 -1.202 0.203 -0.252 0.236 -0.329 -0.386 0.720 0.790 -1.396 -0.793 1.872 -0.152 0.643 0.822 2.153 -0.614 -0.011 -0.209 0.201 -0.581 1.148 0.126 1.108 -0.030 0.318 1.143 0.464 -2.282 1.960 0.248 2.048 -1.707 0.054 0.014 -0.679 0.743 0.634 2.133 -0.417 -0.033 -0.429 -1.420 0.080 0.849 1.635 -1.545 0.286 -1.352 0.125 1.755 0.903 -1.213 -0.509 0.429 -1.395 0.431 -0.681 0.712 0.771 -0.970 -0.803 0.933 -0.156 1.810 0.121 0.029 -1.003 0.974 0.999 0.479 -0.029 -0.077 -0.119 -0.131 -0.643 0.409 1.888 -1.284 0.195 -1.040 -1.335 0.739 -0.236 1.111 0.393 -0.825 0.470 -0.221 -1.450 -0.215 0.321 1.756 0.944 -0.853 1.064 1.967 -1.911 -0.886 -1.160 -2.623 0.306 -2.238 -0.371 1.115 0.082 0.270 0.161 -0.232 -1.482 0.095 0.663 -0.905 -0.569 0.031 -0.155 1.407 1.525 1.204 0.105 -1.049 -0.168 1.386 0.003 1.467 -0.476 -0.450 0.781 -1.093 0.906 0.664 1.583 -0.648 0.792 0.416 0.905 -0.835 -0.863 -0.445 -1.064 0.260

Tabela 6.7: Secvența de valori la portul de ieșire folosită pentru validare în experimentul din problema 6.3.

0.499 0.450 -1.243 0.409 -0.907 -0.324 0.376 0.801 -1.093 2.131 -0.691 -1.270 0.177 -0.737 0.696 -1.113 -1.008 1.435 -0.594 0.070 -0.352 -0.205 -0.142 0.039 0.815 0.496 -0.651 -1.649 1.115 -0.014 -0.354 0.211 0.707 -0.193 -0.768 2.009 -1.931 0.923 -0.692 0.231 -0.117 -1.755 1.062 0.659 -1.296 1.081 -1.085 -0.266 -0.229 -0.595 0.649 -0.297 0.004 0.438 0.781 -0.229 -0.265 1.966 -0.335 -1.664 -0.973 0.094 -0.093 -0.451 0.735 -0.046 0.307 0.367 0.600 -1.292 -0.235 1.467 -0.666 0.917 -0.317 0.976 -1.218 0.834 -0.615 0.346 -0.923 1.084 -0.781 0.935 -0.601 0.379 0.152 0.323 -1.522 2.657 -0.947 2.141 -1.945 1.243 -0.344 -0.631 0.761 0.266 0.960 -0.976 0.048 -0.744 -0.904 0.243 0.416 1.083 -1.403 1.473 -1.528 0.821 1.146 0.443 -0.740 -0.131 0.708 -1.264 0.881 -0.653 0.775 0.248 -1.106 -0.025 0.867 -0.909 1.789 -0.712 0.279 -0.682 0.712 -0.294 0.032 -0.584 -0.323 -0.068 -0.263 -0.666 0.441 1.055 -1.407 0.485 -1.197 -0.763 1.071 -0.893 1.052 0.228 -0.013 0.814 -0.296 -1.030 0.707 0.326 0.920 -0.013 -0.457 1.036 0.636 -2.206 0.432 -0.828 -1.170 0.887 -2.447 1.152 0.162 0.289 -0.014 0.400 0.014 -1.157 0.891 0.481 -0.884 -0.307 0.319 -0.179 1.224 0.553 0.078 -0.482 -0.526 0.408 0.759 0.051 1.660 -0.800 -0.336 0.238 -1.036 1.102 -0.178 0.662 -1.072 1.018 0.026 1.227 -1.140 -0.318 0.035 -0.904

Tabela 6.8: Valorile secvenței de validare la portul de intrare, aU din experimentul prezentat în problema 6.4.

0.999 -0.540 0.575 0.412 0.125 0.844 -1.150 0.729 -0.848 0.538 0.923 -2.236 -0.810 -0.778 0.224 1.371 -1.200 0.976 -1.633 1.440 1.019 0.559 0.454 -0.446 0.510 -0.860 0.703 -0.373 -1.715 0.114 -0.756 1.059 -1.051 -1.186 0.287 -0.237 0.241 -1.465 -1.868 2.096 0.802 -0.132 0.425 2.716 0.447 -0.061 -2.374 0.227 1.385 -1.019 -0.102 -0.238 1.683 0.143 1.393 0.168 1.009 0.740 1.217 1.105 0.574 -0.361 0.863 0.121 -1.035 -1.097 0.985 -0.843 0.388 1.222 -0.217 0.553 0.643 0.383 -0.492 -0.417 0.203 1.486 -0.020 0.013 1.334 1.512 -0.558 -0.396 -0.936 1.099 0.798 -0.315 -0.808 0.606 -1.012 -0.177 -1.157 -2.489 -1.973 -0.402 -1.016 0.376 0.644 0.659 -0.885 0.076 0.966 -1.668 0.469 -0.325 1.801 0.921 -1.252 -0.533 0.795 -0.686 0.304 0.749 -0.073 1.113 0.930 -0.415 -1.528 -0.852 -0.347 1.424 -0.083 -0.638 -0.272 0.735 -1.182 0.327 2.332 -0.167 -1.018 -0.120 0.379 0.680 -1.014 -0.255 1.254 -0.492 -0.301 0.346 0.461 -1.069 1.984 -0.019 -2.241 0.272 2.661 0.495 -0.088 -1.770 0.026 1.265 0.816 0.875 0.877 -0.314 -0.782 1.095 -0.756 0.448 0.308 -1.798 0.302 -1.281 0.717 0.024 -1.782 0.964 -0.278 -0.683 -0.443 -0.204 -2.942 -0.602 -1.073 -0.393 0.478 -0.960 -0.395 -0.349 -0.178 -1.088 -0.399 0.254 1.120 -1.227 -0.441 0.137 0.526 -0.215 0.889 -1.529 1.603 -0.363 0.715 0.669 0.688 0.494 1.010 1.578

Tabela 6.9: Valorile secvenșei de validare la portul de ieșire, aY din experimentul prezentat în problema 6.4.

0.000 0.999 1.457 1.791 2.367 2.628 3.191 2.219 1.248 -0.164 -1.006 -0.202 -1.374 -3.847 -5.991 -6.459 -4.011 -2.010 0.168 0.514 1.278 3.296 5.118 6.102 5.352 4.044 1.714 0.013 -1.202 -3.713 -5.472 -6.308 -4.951 -3.532 -3.543 -3.149 -2.337 -1.178 -1.475 -3.988 -3.788 -1.039 1.361 3.128 6.667 9.617 9.920 5.744 0.712 -1.455 -3.007 -4.104 -4.340 -2.073 0.913 4.285 6.653 8.073 8.697 8.982 9.098 8.487 6.288 4.174 2.413 -0.277 -3.719 -4.948 -5.170 -4.324 -1.451 1.243 3.325 5.037 5.932 5.071 2.791 0.631 0.580 1.151 1.323 2.519 5.032 5.983 4.777 1.843 0.053 0.136 0.250 -0.686 -1.002 -1.732 -2.579 -3.903 -7.116 -11.160 -13.148 -13.128 -10.620 -5.909 -0.449 2.909 4.311 5.434 3.948 1.753 -0.224 0.075 2.091 2.292 0.816 0.148 -0.638 -1.099 -0.302 0.618 2.215 4.377 5.065 2.798 -0.964 -4.177 -4.340 -2.957 -2.077 -1.636 -0.400 -0.270 -0.388 2.102 4.423 4.062 2.368 1.027 0.753 -0.265 -1.686 -1.217 -0.510 -0.461 -0.138 0.750 0.382 1.497 2.952 1.130 -1.221 0.175 2.942 4.451 2.803 0.230 -0.339 0.779 2.689 4.803 5.447 3.868 2.694 1.124 -0.130 -0.450 -2.227 -3.623 -5.005 -4.896 -3.458 -3.530 -2.801 -1.527 -1.152 -1.443 -1.784 -4.710 -7.890 -9.911 -10.273 -8.189 -5.814 -3.864 -2.272 -1.055 -1.170 -1.959 -2.066 -0.481 0.058 -0.631 -1.071 -0.571 -0.059 1.093 0.596 0.967 1.473 2.065 3.094 4.217 4.999 5.803

Tabela 6.10: Valorile vectorului eșantioanelor la portul de intrare, aU din experimentul prezentat în problema 6.6.

1.798 0.688 -0.654 0.762 0.164 0.533 1.173 -0.514 -0.256 0.812 0.536 1.446 -2.485 0.251 -1.433 -0.376 -0.648 1.151 -2.087 0.652 -0.226 -1.168 -0.778 -1.064 0.560 1.510 1.189 0.539 0.257 0.483 -0.253 0.679 -1.341 -0.630 2.012 0.485 -1.063 0.104 0.014 0.842 0.698 -1.045 -0.709 0.711 0.846 -1.145 -0.089 -1.109 0.886 1.010 0.189 -0.400 0.373 0.953 -0.746 0.627 0.083 1.286 0.270 0.656 0.756 -0.673 0.034 0.673 -0.155 0.035 0.377 0.299 -1.237 0.881 -0.300 -1.204 -0.293 0.794 0.837 -0.732 1.206 0.542 0.510 -1.301 -0.297 -0.199 -0.766 -0.761 -1.264 -1.274 0.560 1.799 -0.768 0.015 -0.243 -0.907 1.490 0.166 -0.691 -0.466 1.004 1.300 -2.414 -0.962 -0.243 0.531 -1.884 -1.191 -1.948 -0.791 -0.665 -0.230 1.817 -0.199 0.537 -1.167 -1.329 -0.504 0.524 -0.550 -0.503 1.088 1.175 0.159 0.050 -0.702 -0.130 -0.523 0.804 -1.277 0.601 -0.402 -0.130 -1.487 0.985 -1.148 1.609 -0.577 3.512 0.329 -2.309 0.569 -1.060 -0.223 1.118 -0.557 0.396 -1.606 2.301 0.988 -0.138 0.625 -0.082 0.603 1.148 0.905 -1.422 1.573 -0.436 0.686 0.558 0.537 0.798 -2.188 -1.237 -1.873 -0.154 -0.285 0.952 0.711 0.525 1.707 1.046 -0.532 -1.071 1.842 -1.034 -0.296 -1.203 -1.107 -1.004 1.202 -0.898 -1.846 -0.799 -0.042 -0.431 -1.158 -0.434 -0.373 -0.727 -1.262 -0.623 -0.769 0.231 1.824 0.776 -0.019 1.454 -1.630 -0.839 0.335 -0.746 0.085

Tabela 6.11: Valorile vectorului eșantioanelor la portul de ieșire, aY din experimentul prezentat în problema 6.6.

1.714 1.118 -0.794 0.734 0.508 0.423 1.336 -0.325 -0.591 1.032 0.618 1.403 -2.209 -0.445 -0.681 -0.931 -0.430 1.055 -1.780 -0.106 0.403 -1.630 -0.956 -0.949 0.449 1.866 1.286 0.596 0.331 0.538 -0.204 0.592 -1.096 -1.046 2.146 0.905 -1.415 0.051 0.175 0.755 0.947 -1.157 -0.917 0.842 1.004 -1.105 -0.382 -0.889 0.534 1.391 0.131 -0.458 0.383 1.131 -0.643 0.237 0.393 1.195 0.538 0.505 0.925 -0.628 -0.218 0.828 -0.123 -0.018 0.479 0.341 -1.197 0.688 0.125 -1.530 -0.391 0.922 0.889 -0.629 0.863 0.904 0.262 -1.097 -0.677 -0.021 -0.827 -0.853 -1.183 -1.420 0.600 2.131 -0.607 -0.449 -0.009 -1.144 1.431 0.626 -0.973 -0.565 1.078 1.544 -2.327 -1.610 0.051 0.494 -1.718 -1.684 -1.775 -1.137 -0.548 -0.365 1.794 0.097 0.190 -0.825 -1.788 -0.519 0.519 -0.395 -0.700 1.057 1.516 0.230 -0.053 -0.733 -0.249 -0.484 0.600 -1.029 0.235 0.080 -0.441 -1.359 0.704 -0.665 0.974 0.023 3.022 1.281 -2.974 0.257 -0.603 -0.737 1.387 -0.470 0.029 -1.349 1.840 1.774 -0.557 0.581 0.085 0.385 1.279 1.002 -1.468 1.102 0.117 0.059 1.024 0.369 0.890 -2.024 -1.716 -1.579 -0.539 0.027 0.881 1.069 0.513 1.832 1.336 -0.686 -1.269 1.865 -0.560 -0.954 -0.837 -1.398 -1.039 1.103 -0.612 -2.418 -0.822 0.082 -0.373 -1.294 -0.529 -0.210 -0.770 -1.319 -0.770 -0.615 0.093 2.070 1.106 -0.229 1.399 -1.277 -1.406 0.650 -0.613 -0.183

Tabela 6.12: Valorile vectorului eșantioanelor nedecalate - aN din experimentul prezentat în problema 6.7.

0. -0.35058217 4.7450604 1.11868824 2.18711055 -0.34825469 2.18514436 0.28922371 -0.49590589 2.12844001 2.94132532 2.73650442 2.38779853 6.71131287 5.32118265 7.43705839 5.506694 -0.66525612 -1.69176945 -0.74937275 2.29667039 3.40430134 9.1542454 8.04192021 5.41132554 5.1785614 5.24698391 4.99109124 6.54570972 9.93104547 11.46435569 7.50986988 8.41180801 11.37431239 6.61454985 3.66878591 -0.62467451 -2.28636107 -4.55625603 -8.06519425 -6.4869819 -6.89917791 -7.05516451 -5.91558955 -4.32318456

Tabela 6.13: Valorile vectorului eșantioanelor decalate - aNtau din experimentul prezentat în problema 6.7.

-0.34825469 2.18514436 0.28922371 -0.49590589 2.12844001 2.94132532 2.73650442 2.38779853 6.71131287 5.32118265 7.43705839 5.506694 -0.66525612 -1.69176945 -0.74937275 2.29667039 3.40430134 9.1542454 8.04192021 5.41132554 5.1785614 5.24698391 4.99109124 6.54570972 9.93104547 11.46435569 7.50986988 8.41180801 11.37431239 6.61454985 3.66878591 -0.62467451 -2.28636107 -4.55625603 -8.06519425 -6.4869819 -6.89917791 -7.05516451 -5.91558955 -4.32318456 -3.14785328 -6.93277806 -8.0343955 -5.84213978 -4.1101053

Răspunsuri

Problema 6.1:

$\hat{a} = 1.240$.

Problema 6.2:

$\hat{a} = 0.50$, $\hat{b} = 0.69$.
Dacă zgomotul pe liniile de măsurare este neglijabil, estimatorul celor mai mici pătrate returnează valori foarte apropiate de valorile adevărate deși la portul de intrare se aplică un proces aleator (cu parametri statistici cunoscuți).

Problema 6.3:

Clasa de modele ARX, $y[k] + a \cdot y[k-1] = b \cdot u[k-1] + (e[k])$.
$\hat{a} = 0.465$, $\hat{b} = 0.693$
Coeficientul Spearman r este $\rho = 0.945$, $p_{val} = 0$.
Concluzia analizei ipotezei statistice $H0$ este: secvența măsurată este statistic similară cu secvența estimată. Rezultatul este validat.

Problema 6.4:

Dintre cele trei modele candidate, al doilea model aproximează cel mai bine modelul adevărat al sistemului dinamic deoarece valoarea funcției criteriu este cea mai mică.

Problema 6.5:

În cazul celui de-al doilea model, valoarea maximă pe densitatea de probabilitate empirică a erorilor de aproximare - histograma erorilor - este cea mai mare dintre cele trei modele analizate.
Al doilea model, dintre cele trei modele candidate aproximează cel mai bine modelul adevărat al procesului dinamic.
Se obține aceeași concluzie ca și în problema precedentă.

Problema 6.6:

Modelul propus nu este cea mai bună aproximare a modelului adevărat al procesului dinamic.

Problema 6.7:

(a) $a_{est} = 0.446$, $b_{est} = 0.490$. **(b)** $\alpha^0 = 24.053^0$. **(c)** Erorile de aproximare nu sunt de natura zgomotului alb; conțin o componentă deterministă.

Rezolvări

Rezolvarea problemei 6.1

Etapele rezolvării.
Pe baza expresiei modelului procesului, se scrie sistemul (incompatibil) de ecuații format cu datele prelevate și parametrii estimați ai modelului:

$$\left\{\begin{array}{rcl} y_1 & = & \hat{a} \\ y_2 & = & \hat{a} \\ \vdots & \vdots & \vdots \\ y_N & = & \hat{a} \end{array}\right. \Rightarrow \left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_N \end{array}\right] = \left[\begin{array}{c} 1 \\ 1 \\ \vdots \\ 1 \end{array}\right] \times \hat{a} \Leftrightarrow \mathbf{Y} = \mathbf{X} \times \hat{\theta}.$$

Pseudo-soluția sistemului de ecuații de mai sus este estimatorul celor mai mici pătrate care are expresia generală:

$$\hat{\theta} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1} \times \left[\mathbf{X}^T\mathbf{Y}\right],$$

în care $\mathbf{Y} = [y_1; \ldots; y_N]^T$, $\mathbf{X} = [1; \ldots; 1]^T$ și $\hat{\theta} = \hat{a}$.

Coduri - Aplicația 6.1.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aU=numpy.loadtxt('Cap6Prob61_aSemnal.txt')#preia datele din fisier
```

Reprezentarea grafică a datelor prelevate.

```
fig = pyplot.figure(figsize=(6, 3))
gs = fig.add_gridspec(1, 2,  width_ratios=(7, 2),
                      left=0.1, right=0.9, bottom=0.1, top=0.9,
                      wspace=0.2, hspace=0.2)
ax = fig.add_subplot(gs[0, 0])
ax.plot(aU,linestyle='',marker='o')
ax.grid(True)
ax.set_xlabel('Index_esantioane')
ax.set_ylabel('Amplitudine')
ax_histy = fig.add_subplot(gs[0, 1], sharey=ax)
ax_histy.hist(aU,bins=25,orientation='horizontal')
ax_histy.grid(True)
ax_histy.set_xlabel('Frecventa')
```

Rezultat: în figura(6.1).

Inserarea elementelor în matricele X și Y din formula estimatorului.

```
aY=numpy.reshape(aU,[aU.shape[0],1])# vectorul masuratorilor, aY
aX=numpy.ones_like(aY)#matricea aX - un vector cu toate elementele
    unitare
```

```
aTheta=numpy.matmul(numpy.linalg.inv(numpy.matmul(numpy.transpose(
    aX),aX)),(numpy.matmul(numpy.transpose(aX),aY)))
aTheta
```

```
a_est=aTheta[0][0]
a_est, format(a_est,'5.3f')#valoarea adevarata = 1.25
```

Rezultat: $\hat{a} = 1.24$.

Rezolvarea problemei 6.2

Etapele rezolvării.
Estimarea parametrilor modelului cu estimatorul celor mai mici pătrate.
Se pornește de la expresia modelului funcției de transfer-z în enunțul problemei.
Se folosește operatorul de întârziere cu un pas și se obține ecuația cu diferențe care descrie procesul.

$$y[k] = -a \cdot y[k-1] + b \cdot u[k-1] + (e[k]).$$

Se scrie sistemul (incompatibil) de ecuații format cu datele prelevate și parametrii estimați ai modelului:

$$\begin{cases} y_2 = - \hat{a} \cdot y_1 + \hat{b} \cdot u_1 \\ y_3 = - \hat{a} \cdot y_2 + \hat{b} \cdot u_2 \\ \vdots \quad \vdots \quad \vdots \\ y_N = - \hat{a} \cdot y_{N-1} + \hat{b} \cdot u_{N-1} \end{cases} \Rightarrow \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} -y_1 & u_1 \\ -y_2 & u_2 \\ \vdots & \\ -y_{N-1} & u_{N-1} \end{bmatrix} \times \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} \Leftrightarrow \mathbf{Y} = \mathbf{X} \times \hat{\theta}.$$

Pseudo-soluția sistemului de ecuații de mai sus este estimatorul celor mai mici pătrate care are expresia generală:

$$\hat{\theta} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1} \times \left[\mathbf{X}^T\mathbf{Y}\right].$$

Rezultatul estimării se compară cu datele de catalog (valorile adevărate ale parametrilor modelului).

Coduri - Aplicația 6.2.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aU=numpy.loadtxt('Cap6Prob62_aU.txt')#preia datele din fisier
aY=numpy.loadtxt('Cap6Prob62_aY.txt')#preia datele din fisier
```

Reprezentarea grafică a datelor prelevate.

```
fig = pyplot.figure(figsize=(8, 4))
gs = fig.add_gridspec(2, 2,  width_ratios=(7, 2),
                      left=0.1, right=0.9, bottom=0.1, top=0.9,
                      wspace=0.1, hspace=0.2)
ax1 = fig.add_subplot(gs[0, 0])
ax1.plot(aU,linestyle='',marker='o')
ax1.grid(True)
ax1.set_xlabel('Index esantioane')
ax1.set_ylabel('Amplitudine - aU')
ax1_histy = fig.add_subplot(gs[0, 1], sharey=ax1)
ax1_histy.hist(aU,bins=20,orientation='horizontal')
ax1_histy.grid(True)
```

```
ax1_histy.set_xlabel('Frecventa')

ax2 = fig.add_subplot(gs[1, 0])
ax2.plot(aY,linestyle='',marker='o', color='orange')
ax2.grid(True)
ax2.set_xlabel('Index esantioane')
ax2.set_ylabel('Amplitudine - aY')
ax2_histy = fig.add_subplot(gs[1, 1], sharey=ax2)
ax2_histy.hist(aY,bins=20,orientation='horizontal', color='orange')
ax2_histy.grid(True)
ax2_histy.set_xlabel('Frecventa')
```

Rezultat: în figura (6.2).

Estimarea parametrilor moelului cu estimatorul celor mai mici pătrate.

```
#y[k]=-a*y[k-1]+bU[k-1]
iDimY=aY.shape[0]#numarul de coloane ale vectorului aY = N numarul
    de esantioane
X=numpy.empty([iDimY-1,2])#prototipul matricei X - doua coloane si
    N-1 linii
Y=numpy.empty([iDimY-1,1])#prototipul matricei Y - o coloana si N-1
     linii
for k1 in list(range(iDimY-1)):
    X[k1,0]=-aY[k1]#prima coloana din X, indexul incepe de la 1
        pana la N-1
    X[k1,1]=aU[k1]#a doua coloana din X, indexul incepe de la 1
        pana la N-1
    Y[k1,0]=aY[k1+1]#coloana din Y, indexul incepe de la 2 pana la
        N
```

```
Theta=numpy.matmul(numpy.linalg.inv(numpy.matmul(numpy.transpose(X)
    ,X)),
                   (numpy.matmul(numpy.transpose(X),Y)))#calculul
                       estimatorului celor mai mici patrate
Theta#valorile adevarate a=0.5; b=0.7
```

Rezultat: $\hat{a} = 0.50$, $\hat{b} = 0.69$.

Rezolvarea problemei 6.3

Etapele rezolvării.

- Se adoptă un model din clasa de modele ARX, cu cel mai mic număr de parametri descris prin ecuația cu diferențe:

$$y[k] + a \cdot y[k-1] = b \cdot u[k-1] + (e[k]).$$

- Se implementează algoritmul celor mai mici pătrate folosind pentru identificare secvențele de eșantioane selectate pentru identificare și se estimează valorile parametrilor modelului, $\hat{a}$ și $\hat{b}$ - asemănător cu problema precedentă.

- În continuare, se folosesc secvențele de date selectate pentru validarea modelului. Se calculează valorile estimate ale secvenței la portul de ieșire pe baza secvenței de eșantioane măsurate la portul de intrare și a parametrilor modelului, estimați la pasul precedent care se compară cu eșantioanele măsurate ale secvenței la portul de ieșire.

- Compararea secvențelor de date se poate face prin mai multe metode.
  O metodă simplă, dar intuitivă este testarea ipotezei statistice $H0$. Testul $H0$ oferă informația dacă cele două secvențe sunt necorelate statistic. Testul constă în calculul coeficientului de corelație Spearman-r - notat ρ - care ia valori în intervalul $[-1; 1]$. Dacă $\rho = 0$ atunci secvențele sunt necorelate statistic; dacă $\rho = 1$ secvențele sunt corelate (similare) și dacă $\rho = -1$; atunci secvențele sunt disimilare (simiare dar în sens de variație invers).

- Dacă rezultatul identificării nu poate fi validat la pasul precedent, atunci se alege o altă clasă de modele - cu un număr mai mare de parametri și, se reia procedura de analiză.

Coduri - Aplicația 6.3.
Se apelează modulele necesare.

```
import numpy
import sympy
from scipy import stats
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aUid=numpy.loadtxt('Cap6Prob63_aUid.txt')#preia datele pentru
    identificare din fisier
aYNid=numpy.loadtxt('Cap6Prob63_aYNid.txt')#preia datele pentru
    identificare din fisier
aUval=numpy.loadtxt('Cap6Prob63_aUval.txt')#preia datele pentru
    validare din fisier
aYNval=numpy.loadtxt('Cap6Prob63_aYNval.txt')#preia datele pentru
    validare din fisier
```

Reprezentarea grafică a datelor prelevate.

```
fig = pyplot.figure(figsize=(8, 4))
gs = fig.add_gridspec(2, 2,  width_ratios=(7, 2),
                      left=0.1, right=0.9, bottom=0.1, top=0.9,
                      wspace=0.1, hspace=0.2)
ax1 = fig.add_subplot(gs[0, 0])
ax1.plot(aUid,linestyle='',marker='o')
ax1.grid(True)
ax1.set_xlabel('Index esantioane')
ax1.set_ylabel('Amplitudine - aUid')
ax1_histy = fig.add_subplot(gs[0, 1], sharey=ax1)
ax1_histy.hist(aUid,bins=20,orientation='horizontal')
ax1_histy.grid(True)
ax1_histy.set_xlabel('Frecventa')

ax2 = fig.add_subplot(gs[1, 0])
ax2.plot(aYNid,linestyle='',marker='o', color='orange')
ax2.grid(True)
ax2.set_xlabel('Index esantioane')
ax2.set_ylabel('Amplitudine - aYNid')
ax2_histy = fig.add_subplot(gs[1, 1], sharey=ax2)
ax2_histy.hist(aYNid,bins=20,orientation='horizontal', color='
    orange')
ax2_histy.grid(True)
ax2_histy.set_xlabel('Frecventa')
```

Rezultat: în figura(6.3).

Estimarea parametrilor modelului cu primul set de date.

```
iDimY=aUid.shape[0]#numarul de coloane ale vectorului aY = N
    numarul de esantioane
X=numpy.empty([iDimY-1,2])#prototipul matricei X - doua coloane si
    N-1 linii
Y=numpy.empty([iDimY-1,1])#prototipul matricei Y - o coloana si N-1
     linii
for k1 in list(range(iDimY-1)):
    X[k1,0]=-aYNid[k1]#prima coloana din X, indexul incepe de la 1
        pana la N-1
    X[k1,1]=aUid[k1]#a doua coloana din X, indexul incepe de la 1
        pana la N-1
    Y[k1,0]=aYNid[k1+1]#coloana din Y, indexul incepe de la 2 pana
        la N
Theta=numpy.matmul(numpy.linalg.inv(numpy.matmul(numpy.transpose(X)
    ,X)),
                   (numpy.matmul(numpy.transpose(X),Y)))#calculul
                       estimatorului celor mai mici patrate
Theta
```

Rezultat: $\hat{a} = 0.465$ și $\hat{b} = 0.693$.

Validarea rezultatului cu cel de-al doilea set de date.

```
dAest=Theta[0,0]#valoarea estimata a parametrului a
dBest=Theta[1,0]#valoarea estimata a parametrului b
aYest=numpy.empty_like(aYNval)#prototipul vectorului valorilor
    estimate
aYest[0]=0#valoarea initiala
for k1 in list(range(iDimY-1)):
    aYest[k1+1]=-dAest*aYest[k1]+dBest*aUval[k1]
```

```
rho,pval=stats.spearmanr(aYNval,aYest)#calculul coeficientului de
    similaritate Spearman R.
rho, pval
```

Rezultat: $\rho = 0.945$ și $p_{val} = 0$.

Interpretarea rezultatului testului.
Valoarea coeficientului de corelație ρ este foarte aproape de 1 ceea ce indică similaritate (în sens statistic) între cele două secvențe de eșantioane.
Rezultă că rezultatul identificării este validat.

Rezolvarea problemei 6.4

Etapele rezolvării.

- Pentru fiecare model candidat se calculează erorile de aproximare. Acestea reprezintă diferența dintre valorile măsurate la portul de ieșire și valorile estimate pe baza secvenței la portul de intrare și a parametrilor modelului:

$$\epsilon_k = y_k - \hat{y}_k, \forall k = 1 \dots N.$$

- În continuare se calculează valorile funcției criteriu, i.e. suma pătratelor valorilor erorilor de aproximare:

$$J = \sum_{k=1}^{N} \epsilon_k^2 = \epsilon^T \times \epsilon.$$

- Cu cât modelul aproximează mai bine modelul adevărat al procesului dinamic cu atât valoarea funcției criteriu corespunzătoare este mai mică.
  Prin urmare, se compară valorile funcției criteriu corespunzătoare celor trei modele candidate și se alege modelul căruia îi corespunde valoarea cea mai mică.

- Deoarece, în aplicație vom folosi funcțiile modulului *scipy.signal*, se rescriu ecuațiile cu diferențe care descriu cele trei modele candidate sub forma funcțiilor de transfer-z. Rezultă expresiile:

$$\begin{aligned} H_1(z) &= \frac{z + 0.3}{z^2 - 1.501z + 0.55}; \\ H_2(z) &= \frac{z + 0.551}{z^2 - 1.491z + 0.707}; \\ H_3(z) &= \frac{z^2 + 0.435z + 0.25}{z^3 - 1.501z^2 + 0.55z + 0.152}. \end{aligned}$$

Coduri - Aplicația 6.4.
Se apelează modulele necesare.

```
import numpy
import sympy
from scipy import signal
from scipy import stats
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aTimp=numpy.loadtxt('Cap6Prob64_aTimp.txt')#preia datele din fisier
    - secventa esantioanelor de timp
aU=numpy.loadtxt('Cap6Prob64_aU.txt')#preia datele din fisier -
   secventa la portul de intrare
aY=numpy.loadtxt('Cap6Prob64_aY.txt')#preia datele din fisier -
   secventa la portul de iesire
```

Reprezentarea grafică a datelor prelevate.

```
fig = pyplot.figure(figsize=(8, 4))
gs = fig.add_gridspec(2, 2,  width_ratios=(7, 2),
                      left=0.1, right=0.9, bottom=0.1, top=0.9,
                      wspace=0.1, hspace=0.2)
ax1 = fig.add_subplot(gs[0, 0])
ax1.plot(aTimp,aU,linestyle='',marker='o')
ax1.grid(True)
ax1.set_xlabel('Timpul')
ax1.set_ylabel('Amplitudine - aU')
ax1_histy = fig.add_subplot(gs[0, 1], sharey=ax1)
ax1_histy.hist(aU,bins=20,orientation='horizontal')
ax1_histy.grid(True)
ax1_histy.set_xlabel('Frecventa')

ax2 = fig.add_subplot(gs[1, 0])
ax2.plot(aTimp,aY,linestyle='',marker='o', color='orange')
ax2.grid(True)
ax2.set_xlabel('Timpul')
ax2.set_ylabel('Amplitudine - aY')
ax2_histy = fig.add_subplot(gs[1, 1], sharey=ax2)
ax2_histy.hist(aY,bins=20,orientation='horizontal', color='orange')
ax2_histy.grid(True)
ax2_histy.set_xlabel('Frecventa')
```

Rezultat:

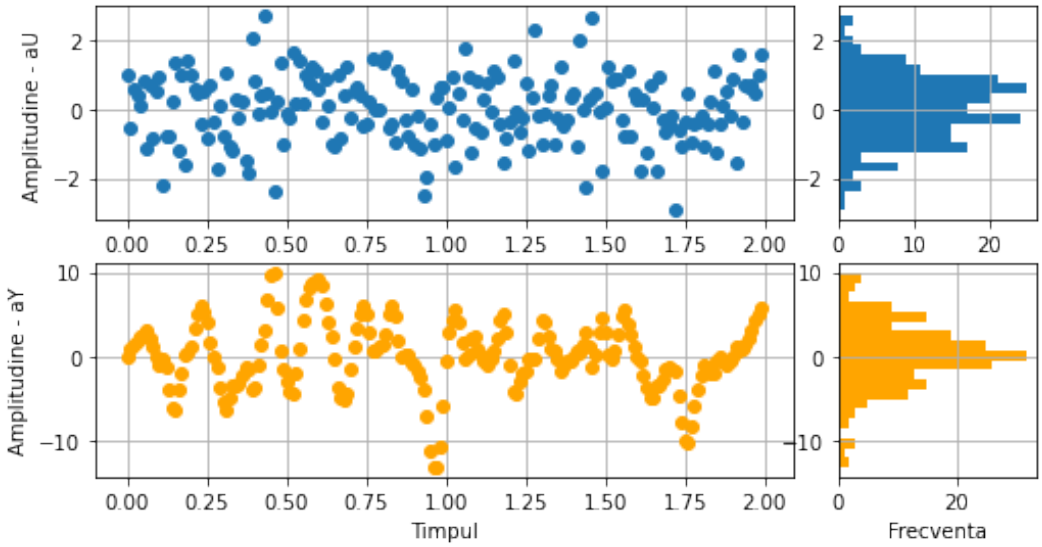


Figura 6.5: Reprezentarea grafică a datelor prelevate în experimentul din problema 6.4.

Estimarea valorii funcției criteriu - Primul model.

```
#H1(z)=(z+0.3)/(z^2-1.501z+0.55) functia de transfer echivalenta
    dTe=1
H1_num=[1, 0.3]#numaratorul functiei de transfer
H1_den=[1, -1.501, 0.55]#numitorul functiei de transfer
```

```
H1=signal.TransferFunction(H1_num, H1_den,dt=1)#functia de transfer
    a primului model
tout,aY1est=signal.dlsim(H1,aU)#estimarea esantioanelor raspunsului
    primului model
aEps1=aY-numpy.transpose(aY1est[:,0])
dJ1=numpy.matmul(numpy.transpose(aEps1),aEps1)
dJ1, format(dJ1,'5.3f')
```

Rezultat: $J_1 = 5246.494$.

Estimarea valorii funcției criteriu - Al doilea model.

```
#H2(z)=(z+0.551)/(z^2-1.491z+0.707)
H2_num=[1, 0.551]#numaratorul functiei de transfer
H2_den=[1, -1.491, 0.707]#numitorul functiei de transfer
H2=signal.TransferFunction(H2_num, H2_den,dt=1)#functia de transfer
    a celui de-al doilea model
tout,aY2est=signal.dlsim(H2,aU)#estimarea esantioanelor raspunsului
    celui de-al doilea model
aEps2=aY-numpy.transpose(aY2est[:,0])#calcul erorilor de aproximare
dJ2=numpy.matmul(numpy.transpose(aEps2),aEps2)#calculul valorii
    functiei criteriu
dJ2,format(dJ2,'5.3f')
```

Rezultat: $J_2 = 20.201$.

Estimarea valorii funcției criteriu - Al treilea model.

```
H3_num=[1, 0.435, 0.25]#numaratorul functiei de transfer
H3_den=[1, -1.501, 0.55, 0.152]#numitorul functiei de transfer
H3=signal.TransferFunction(H3_num, H3_den,dt=1)#functia de transfer
    a celui de-al treilea model
tout,aY3est=signal.dlsim(H3,aU)#estimarea esantioanelor raspunsului
    celui de-al treilea model
aEps3=aY-numpy.transpose(aY3est[:,0])#calcul erorilor de aproximare
dJ3=numpy.matmul(numpy.transpose(aEps3),aEps3)#calculul valorii
    functiei criteriu
dJ3,format(dJ3,'5.3f')
```

Rezultat: $J_3 = 1289.524$.

CONCLUZIE:
Valoarea minimă a funcției criteriu corespunde celui de-al doilea model; deci, dintre cele trei modele candidate, al doilea model este cea mai bună aproximare a modelului adevărat.

Rezolvarea problemei 6.5

Etapele rezolvării.
Metoda verosimilității maxime constă în analiza densității de probabilitate empirice (histograma) erorilor de aproximare ale identificării.
Principiul metodei este următorul: fiind date un set de modele candidate și histogramele erorilor de aproximare corespunzătoare. Modelul care aproximează cel mai bine modelul adevărat al procesului dinamic este acela căruia îi corespunde histograma cu valoarea maximă cea mai mare.

- Pe baza secvențelor de valori ale eșantioanelor la porturile de intrare/ieșire, se calculează erorile de aproximare pentru cele trei modele candidate, asemănator cu problema precedentă.

- Se calculează histogramele erorilor de aproximare pentru fiecare dintre modele.

- Se compară valorile maxime pe histograme și se implementează principiul verosimilității maxime.

Coduri - Aplicația 6.5.
Se apelează modulele necesare.

```
import numpy
import sympy
from scipy import signal
from scipy import stats
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aTimp=numpy.loadtxt('Cap6Prob64_aTimp.txt')#preia datele din fisier
    - secventa esantioanelor de timp
aU=numpy.loadtxt('Cap6Prob64_aU.txt')#preia datele din fisier -
    secventa la portul de intrare
aY=numpy.loadtxt('Cap6Prob64_aY.txt')#preia datele din fisier -
    secventa la portul de iesire
```

Estimarea erorilor de aproximare - Primul model

```
#H1(z)=(z+0.3)/(z^2-1.501z+0.55) functia de transfer echivalenta
    dTe=1
H1_num=[1, 0.3]#numaratorul functiei de transfer
H1_den=[1, -1.501, 0.55]#numitorul functiei de transfer
H1=signal.TransferFunction(H1_num, H1_den,dt=1)#functia de transfer
    a primului model
tout,aY1est=signal.dlsim(H1,aU)#estimarea esantioanelor raspunsului
    primului model
aEps1=aY-numpy.transpose(aY1est[:,0])#calculeaza vectorul erorilor
    de aproximare
```

Estimarea erorilor de aproximare - Al doilea model

```
#H2(z)=(z+0.551)/(z^2-1.491z+0.707)
H2_num=[1, 0.551]#numaratorul functiei de transfer
H2_den=[1, -1.491, 0.707]#numitorul functiei de transfer
H2=signal.TransferFunction(H2_num, H2_den,dt=1)#functia de transfer
    a celui de-al doilea model
tout,aY2est=signal.dlsim(H2,aU)#estimarea esantioanelor raspunsului
    celui de-al doilea model
aEps2=aY-numpy.transpose(aY2est[:,0])#calcul erorilor de aproximare
```

Estimarea erorilor de aproximare - Al treilea model

```
#H3(z)=(z^2+0.435z+0.25)/(z^3-1.501z^2+0.55z+0.152)
H3_num=[1, 0.435, 0.25]#numaratorul functiei de transfer
H3_den=[1, -1.501, 0.55, 0.152]#numitorul functiei de transfer
H3=signal.TransferFunction(H3_num, H3_den,dt=1)#functia de transfer
    a celui de-al treilea model
tout,aY3est=signal.dlsim(H3,aU)#estimarea esantioanelor raspunsului
    celui de-al treilea model
aEps3=aY-numpy.transpose(aY3est[:,0])#calcul erorilor de aproximare
```

Calculul histogramelor (densitățile de probabilitate empirice) ale erorilor de aproximare.

```
fig, ax=pyplot.subplots(nrows=1,ncols=1)
ax.hist(aEps1,bins=20,density=True,label='Modelul 1')
ax.hist(aEps2,bins=20,density=True,label='Modelul 2')
ax.hist(aEps3,bins=20,density=True, label='Modelul 3')
ax.grid(True)
ax.legend()
ax.set_xlabel('Bins')
ax.set_ylabel('Density')
```

Rezultat:

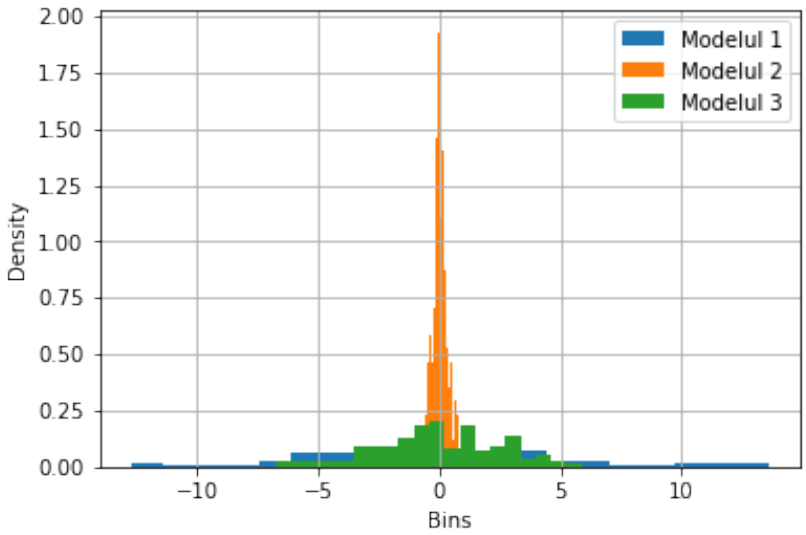


Figura 6.6: Histogramele erorilor de aproximare pentru setul de trei modele candidate din experimentul prezentat în problema 6.5.

Rezolvarea problemei 6.6

Etapele rezolvării.
Modelul care reprezintă cea mai bună aproximare unui proces dinamic se caracterizează prin aceea că pentru orice set de două secvențe ale proceselor la porturile de intrare/ieșire, erorile de aproximare sunt de natura zgomotului alb.
Prin urmare trebuie să testăm pe baza datelor din problemă ipoteza că erorile de aproximare ale modelului propus sunt de natura zgomotului alb. Testarea se poate face cu ajutorul funcției de autocorelație a erorilor de aproximare.
Ipoteza se confirmă dacă reprezentarea grafică a funcției de autocorelație are aspectul graficului funcției de autocorelație a zgomotului alb - asemănător cu graficul impulsului Dirac.

Coduri - Aplicația 6.6.
Se apelează modulele necesare.

```
import numpy
import sympy
from scipy import signal
from scipy import stats, correlate
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor din experiment.

```
aU=numpy.loadtxt('Cap6Prob66_aU.txt')#preia datele din fisier -
    secventa la portul de intrare
aY=numpy.loadtxt('Cap6Prob66_aYN.txt')#preia datele din fisier -
    secventa la portul de iesire
```

Reprezentarea grafică a datelor prelevate.

```
fig = pyplot.figure(figsize=(8, 4))
gs = fig.add_gridspec(2, 2,  width_ratios=(7, 2),
                      left=0.1, right=0.9, bottom=0.1, top=0.9,
                      wspace=0.1, hspace=0.2)
ax1 = fig.add_subplot(gs[0, 0])
ax1.plot(aU,linestyle='',marker='o')
ax1.grid(True)
ax1.set_xlabel('Index esantioane')
ax1.set_ylabel('Amplitudine - aU')
ax1_histy = fig.add_subplot(gs[0, 1], sharey=ax1)
ax1_histy.hist(aU,bins=20,orientation='horizontal')
ax1_histy.grid(True)
ax1_histy.set_xlabel('Frecventa')

ax2 = fig.add_subplot(gs[1, 0])
ax2.plot(aY,linestyle='',marker='o', color='orange')
ax2.grid(True)
ax2.set_xlabel('Index esantioane')
ax2.set_ylabel('Amplitudine - aYN')
ax2_histy = fig.add_subplot(gs[1, 1], sharey=ax2)
```

```
ax2_histy.hist(aY, bins=20, orientation='horizontal', color='orange')
ax2_histy.grid(True)
ax2_histy.set_xlabel('Frecventa')
```

Rezultat:

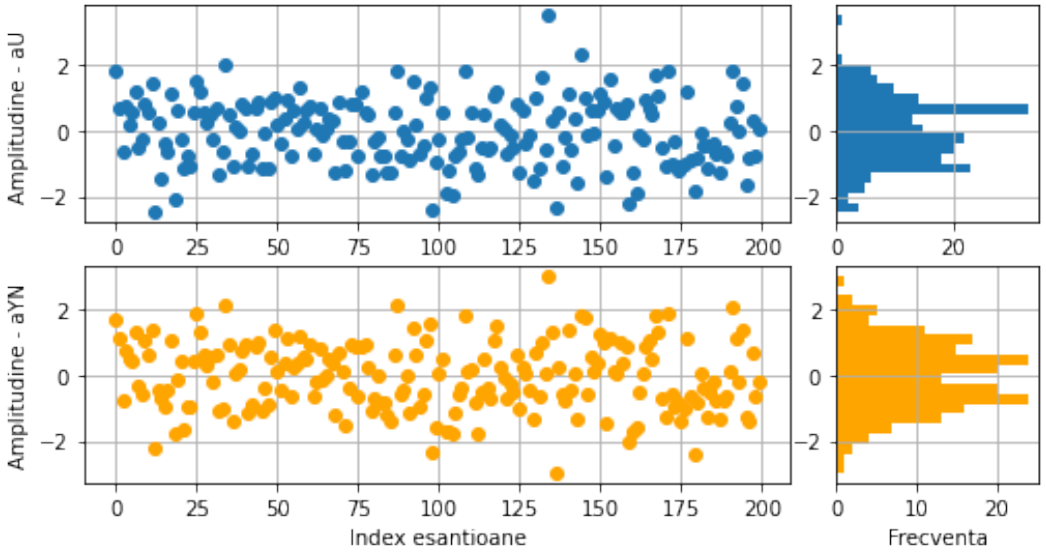


Figura 6.7: Secvențele de eșantioane prelevate în experimentul din problema 6.6.

Estimarea erorilor de aproximare.

```
#H(z)=(z+b1)/(z^2+a1*z+a2) functia de transfer echivalenta dTe=1
a1=0.74#valoarea estimata a parametrului a1
a2=0.02#valoarea estimata a parametrului a2
b1=0.14#valoarea estimata a parametrului b1
H_num=[1,b1]#numaratorul functiei de transfer
H_den=[1, a1, a2]#numitorul functiei de transfer
H=signal.TransferFunction(H_num, H_den,dt=1)#functia de transfer a
    primului model
tout,aYest=signal.dlsim(H,aU)#estimarea esantioanelor raspunsului
    primului model
aEps=aY-numpy.transpose(aYest[:,0])#calculeaza vectorul erorilor de
    aproximare
```

Calculul și reprezentarea grafică a funcției de autocorelație a erorilor de aproximare.

```
aEpsCorr=numpy.correlate(aEps,aEps, mode='full')
```

```
fig, ax=pyplot.subplots(nrows=1,ncols=1)
ax.plot(aEpsCorr)
ax.grid(True)
#ax.legend()
ax.set_xlabel('Tau')
ax.set_ylabel('Functia de autocorelatie')
```

Rezultat:

INTERPRETAREA REZULTATELOR.

Aspectul reprezentării grafice a funcției de autocorelație nu este de forma reprezentării

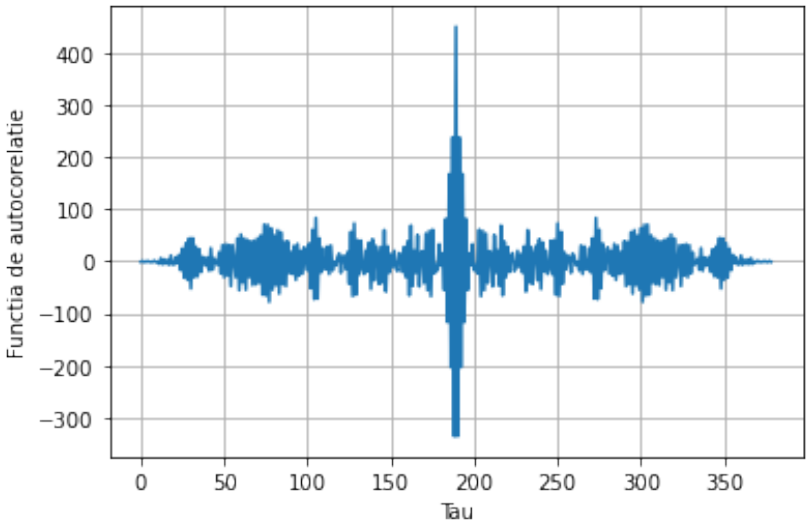


Figura 6.8: Reprezentarea grafică a funcției de autocorelație a erorilor de aproximare pentru experimentul din problema 6.6.

grafice a funcției de autocorelație a zgomotului alb. Prin urmare, modelul propus nu este cea mai buna aproximare a modelului adevărat al procesului dinamic.

Observații.

- Compararea reprezentării grafice a funcției de autocorelație a erorilor de aproximare cu reprezentarea grafică generică a zgomotului alb oferă o informație calitativă asupra rezultatului bazată pe experiența experimentatorului.

- Informația cantitativă poate fi obținută dacă se calculează coeficienții de corelație dintre eșantioanele secvenței erorilor de aproximare și eșantioanele aceleiași secvențe la care indexul eșantioanelor este decalat cu τ pași (maximum 10% din numărul total de eșantioane din secvență).
  Criteriul pe baza căruia se confirmă ipoteza similitudinii cu zgomotul alb este: valorile coeficienților de corelație situați pe subdiagonalele adiacente diagonalei principale să fie zero (foarte aproape de zero).

Următoarele coduri permit calculul coeficienților de corelație pentru $\tau = 7$.

```
Tau=7#valoarea deplasarii in domeniul timpului
aEps1=aEps[0:aEps.shape[0]-Tau]#secventa de esantioane nedeplasata
aEps1Tau=aEps[Tau:aEps.shape[0]]#secventa de esantioane deplasata
    cu tau
aCorr=numpy.corrcoef(aEps1,aEps1Tau)#calculeaza matricea
    esantioanelor
#functiei de autocorelatie
aCorr, sympy.print_latex(aCorr)
```

Rezultat: $\begin{bmatrix} 1. & -0.12812138 \\ -0.12812138 & 1. \end{bmatrix}$.

Rezolvarea problemei 6.7

Etapele rezolvării.

- Erorile de aproximare oferă informația privind calitatea modelului estimat. Prezența componentei deterministe în secvența erorilor de aproximare indică faptul că ipoteza pe baza căreia a fost făcută estimarea se confirmă doar în parte.

- Spre deosebire de abordarea din problema 6.6, care permite evaluarea calitativă a rezultatului estimării, analiza de corelație a erorilor de aproximare permite evaluarea mai precisă - cantitativă - a rezultatului estimării.

- Se calculează dreapta de regresie dintre secvența erorilor de aproximare și secvența erorilor de aporximare deplasată cu o valoare τ - cel mult 10% din lungimea secvenței.

- Ipoteza statistică $H0$ privind absența corelațiilor statistice dintre cele două secvențe se testează prin evaluarea valorii unghiului α dintre dreapta de regresie și axa orizontală. Dacă unghiul $alpha = 45^0$ atunci cele două secvențe sunt statistic similare și deci secvențele sunt corelate statistic. Dacă unghiul $alpha = 0^0$

Coduri - Aplicația 6.7.
Se apelează modulele necesare.

```
import numpy
import sympy
from matplotlib import pyplot
from sympy import*
sympy.init_printing()
```

Preluarea datelor prelevate.

```
aN=numpy.loadtxt('Cap6Problema67_aN.txt')#preia datele din fisier
aNtau=numpy.loadtxt('Cap6Problema67_aNtau.txt')#preia datele din
    fisier
```

```
fig,ax=pyplot.subplots(nrows=1, ncols=1)
ax.scatter(aN,aNtau)
ax.set_xlabel('aN_[s]')
ax.set_ylabel('aNtau')
ax.grid(True)
```

Rezultat:

```
S1=numpy.sum(aN*aN)#calculul sumei S1
S2=numpy.sum(aN)#calculul sumei S2
S3=numpy.sum(aN*aNtau)#calculul sumei S3
S4=numpy.sum(aNtau)#calculul sumei S4
N=numpy.size(aN)#numarul de esantioane
S1,S2,S3,S4,N
```

```
a_est=sympy.symbols('a_est', real=True)#definitia variabilei a_est
b_est=sympy.symbols('b_est', real=True)#definitia variabilei b_est
expr=sympy.linsolve([S1*a_est+S2*b_est-S3,S2*a_est+b_est*N-S4],(
    a_est,b_est))#rezolvarea sistemului de ecuatii liniare
```

```
a_est=list(expr)[0][0]#preia valoarea variabilei din structura de
    date
b_est=list(expr)[0][1]#preia valoarea variabilei din structura de
    date
a_est, b_est, format(a_est,'5.3f'), format(b_est,'5.3f')
```

Rezultat: $a_{est} = 0.446$, $b_{est} = 0.490$.

```
aNest=a_est*aN+b_est
fig,ax=pyplot.subplots(nrows=1, ncols=1)

ax.scatter(aN,aNest,color='orange',label='Valori estimate aNest vs
    aN')
ax.plot(aN,aNest, color='orange',label='Valori estimate aNest vs aN
    ')

ax.scatter(aN,aN, color='green', label='Valori masurate aN vs aN')
ax.plot(aN,aN, color='green',label='Valori')

ax.scatter(aN,aNtau, color='blue',marker='o', label='Valori
    masurate aNtau vs aN')

ax.set_xlabel('aN')
ax.set_ylabel('aNtau')
ax.grid(True)
ax.legend()
```

Rezultat:

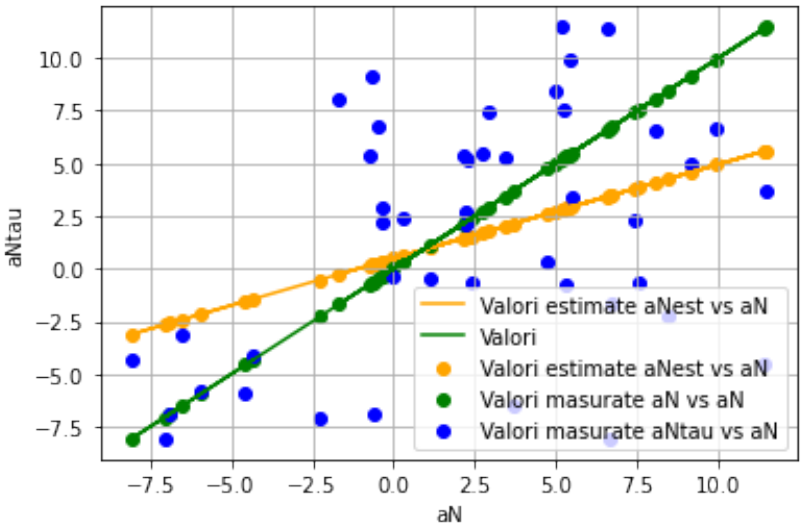


Figura 6.9: Poziționarea în plan a dreptei de regresie eșantioanelor erorilor de aproximare din problema 6.7.

Calculul unghiului dintre axa orizontală și direcția dreptei de regresie.

```
alpha=(sympy.atan(a_est)*180/sympy.pi).evalf()
alpha, format(alpha,'5.3f')
```

Rezultat: $\alpha^0 = 24.053^0$.

Interpretarea rezultatului
$\alpha^0 \neq 0$ indică prezența unei componentei deterministe în procesul aleator; $\alpha^0 \neq 45^0$ indică absența similarității statistice între cele două secvențe de eșantioane.

Bibliografie

[1] Soderstrom T., Stoica P. (1989): *System Identification*, Prentice Hall International.

[2] Dănilă A. (2013): *Modelarea și Identificarea sistemelor dinamice*, Editura Universității Transilvania din Brașov.

[3] Mihoc, Gh., Micu N. (1980): *Teoria probabilităților și Statistică matematică*, Editura Didactică și Pedagogică, București.

[4] Săvescu M., Petrescu, T., Ciochina, S. (1980): *Semnale, Circuite și Sisteme. Probleme* Editura Didactică și Pedagogică, București, pp. 187 - 197.

[5] Besekerski V. A., Paltov, I. P. Fabricant, E. A. Fedorov, S. M. Cinaev P. I. (1984): *Teoria Reglării automate. Culegere de probleme* Editura Tehnică, București, pp. 158 - 178.

[6] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, Kumar A, Ivanov S, Moore JK, Singh S, Rathnayake T, Vig S, Granger BE, Muller RP, Bonazzi F, Gupta H, Vats S, Johansson F, Pedregosa F, Curry MJ, Terrel AR, Roučka Š, Saboo A, Fernando I, Kulal S, Cimrman R, Scopatz A. (2017): *SymPy: symbolic computing in Python.* *PeerJ Computer Science* 3:e103 https://doi.org/10.7717/peerj-cs.103

Dănilă Adrian este absolvent al Universităţii Transilvania din Braşov, promoţia 1985, în specialitatea Electrotehnică şi doctor în domeniul Inginerie Electrică.
Din anul 2006, Dănilă este titularul cursului *Modelarea şi Identificarea sistemelor* la Departamentul de Automatică şi Tehnologia informaţiei, Facultatea de Inginerie electrică şi Ştiinţa calculatoarelor.

Editura Universităţii *Transilvania* din Braşov