Algorytmy i metody optymalizacji (AMO)
Projekt II, zestaw AK, nr 2

Arkadiusz Piórkowski

Zadanie

Pewien obiekt dynamiczny jest opisany równaniami:

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = u(t) - x_2(t),$$

przy warunkach początkowych: $x_1(0) = 0,\ x_2(0) = -1.$

Znaleźć sterowanie $u(t),\ 0 \leqslant u(t) \leqslant 1,$ minimalizujące funkcjonał jakości:

$$J(x,u) = \int_0^1 [x_1^2(t) + x_2^2(t) + 0.005 \cdot u^2(t)]\, dt$$

przy ograniczeniu na stan:

$$x_2(t) \leqslant 8(t-0.5)^2 - 0.5,\ 0 \leqslant t \leqslant 1$$

Zastosować przedziałami stałą (schodkową) reprezentację funkcji $u(t)$ i $x_1(t),\ x_2(t)$, dzieląc cały przedział sterowania na $N = 50$ podprzedziałów.

Traktując wszystkie zmienne $u(t)$ i $x_1(t),\ x_2(t)$ jako niezależne zmienne decyzyjne, przedstawiono zadanie sterowania optymalnego jako zadanie programowania nieliniowego. Dane zadanie rozwiązano przy pomocy programu AMPL i Matlab.

Spis treści

Zadanie 1
1. Przekształcenie równań różniczkowych do ograniczeń równościowych.....3
2. Przekształcenie wskaźnika jakości typu Bolzy.....4
3. Rozwiązanie zadania optymalizacji.....5
1. Matlab.....5
1. Funkcja celu - objective_fun.m.....5
2. Ograniczenia - nonlcon.m.....6
3. Kod główny programu – main.m.....6
2. AMPL.....7
1. Model – project2.mod.....7
2. Dane – project2.dat.....8
3. Wykonywalny – project2.run.....8
4. Uzyskane wyniki.....9

1. Przekształcenie równań różniczkowych do ograniczeń równościowych

Stosując metodę Eulera przekształcono równania różniczkowe (równania stanu) do ograniczeń równościowych.

Przyjmując:

$$\dot{x}(t)=\frac{dx(t)}{dt}\approx\frac{x(t+\Delta)-x(t)}{\Delta},$$

przekształcono równania różniczkowe:

$$\dot{x}(t)=f(x(t),u(t),t),\ \ t\in[t_{0,}t_f],$$

w układy równań:

$$x(t_{k+1})-x(t_k)=\Delta\cdot f(x(t_k),u(t_k),t_k),\ \ k=0,1,\ldots,N-1,\ \ \Delta=\frac{t_f-t_0}{N},\ \ t_k=t_0+k\cdot\Delta,$$

które w prostszej notacji mają formę:

$$x_{k+1}-x_k=\Delta\cdot f(x_k,u_k,k),\ \ k=0,1,\ldots,N-1.$$

Dla danego obiektu otrzymano następujące układy równań określające ograniczenia równościowe zadania:

$$\begin{aligned}x_1(k+1)-x_1(k)&=\Delta x_2(k)\\ x_2(k+1)-x_2(k)&=\Delta(u(k)-x_2(k)).\end{aligned}$$

Należy pamiętać, iż warunki początkowe również określają ograniczenia równościowe:

$$\begin{aligned}x_1(0)&=0\\ x_2(0)&=-1.\end{aligned}$$

Dodatkowo należy w tej samej notacji przedstawić ograniczenia nierównościowe:

$$x_2(k)\leqslant 8(k\cdot\Delta-0.5)^2-0.5,\ \ k=0,1,\ldots,N-1.$$

2. Przekształcenie wskaźnika jakości typu Bolzy

Wskaźnik jakości typu Bolzy reprezentuje funkcję celu, którą w danym zadaniu minimalizujemy:

$$J(x,u)=\int_{t_0}^{t_f} g(x(t),u(t),t)dt+g_f(x(t_f)).$$

Dany wskaźnik przekształcono w sumę:

$$J(x,u)=\Delta\sum_{k=0}^{N-1} g(x_k,u_k,t_k)+g_f(x_N)$$

Dla danego zadania funkcja celu ma postać:

$$J(x,u)=\Delta\sum_{k=0}^{N-1} (x_1^2(k)+x_2^2(k)+0.005\cdot u^2(k))$$

3. Rozwiązanie zadania optymalizacji

Dane zadanie programowania nieliniowego jest następujące:

$$\min_{x_1, x_2, u}\left(\Delta \sum_{k=0}^{N-1} \left(x_1^2(k)+x_2^2(k)+0.005\cdot u^2(k)\right)\right)$$

przy ograniczeniach:

$$x_1(0)=0$$
$$x_2(0)=-1$$
$$x_1(k+1)-x_1(k)=\Delta x_2(k)$$
$$x_2(k+1)-x_2(k)=\Delta(u(k)-x_2(k))$$
$$x_2(k)\leqslant 8(k\ \cdot\Delta-0.5)^2-0.5$$

dla $k=0,1,\ldots,N-1$.

Wymienione zadanie optymalizacji rozwiązano za pomocą programów AMPL i Matlab. Poniżej zamieszczono kod wykorzystywany w programach.

1. Matlab

Zadanie w programie Matlab rozwiązano przy pomocy solwera `fmincon` z Optimization Toolbox-a, które rozwiązuje nieliniowe zadania z ograniczeniami. Solwer ten umożliwia wykorzystanie kilku algorytmów: `'interior-point'`, `'trust-region-reflective'`, `'sqp'`, `'sqp-legacy'`, `'active-set'`. Do rozwiązania danego zadania wybrano algorytm `'sqp'`. Polecenie `fmincon` umożliwia wybór wersji rozwiązania: gradientową lub bezgradientową. Umożliwienie wprowadzenie gradientu uzyskuje się poprzez ustawienie opcji `'SpecifyObjectiveGradient',true`. Do rozwiązania wykorzystano zarówno wersję gradientową jak i bezgradientową, rozpoczynając optymalizację z różnych punktów. Gradient obliczono analitycznie.

1. Funkcja celu - `objective_fun.m`

```
function [ obj, grad ] = objective_fun( x )
global N D
%D - Delta
%x(1,:)=x1
%x(2,:)=x2
%x(3,:)=u
%objective function
obj=0;
for k=1:N
    obj = obj + x(1,k)^2 + x(2,k)^2 + 0.005*x(3,k)^2;
end
obj=obj*D;
%gradient
if nargout > 1
    grad = zeros(3,N);
    for k=1:N
        grad(1,k) = D*2*x(1,k);
        grad(2,k) = D*2*x(2,k);
        grad(3,k) = D*2*0.005*x(3,k);
```

```
    end
end
end
```

2. Ograniczenia - `nonlcon.m`

```
function [ c, ceq ] = nonlcon( x )
global N x10 x20 D
%D - Delta
%x(1,:)=x1
%x(2,:)=x2
%x(3,:)=u
% equality constraints
ceq(1)=x(1,1)-x10;
ceq(2)=x(2,1)-x20;
for k=1:N-1
    ceq(k*2+1)=x(1,k+1)-x(1,k)-D*x(2,k);
    ceq(k*2+2)=x(2,k+1)-x(2,k)-D*(x(3,k)-x(2,k));
end
% nonlinear inequality constraints
for i=1:N
    c(i)=x(2,i)-(8*((i-1)*D-0.5)^2-0.5);
end
end
```

3. Kod główny programu – `main.m`

```
%%AMO, Projket II zestaw AK, nr 2, Arkadiusz Piorkowski 259079
clear all;
global N D x10 x20
%pass data
N=50;                %number of samples
t0=0;                %start time
tf=1;                %end time
D=(tf-t0)/N;         %Delta
x10=0;               %x1(0)
x20=-1;              %x2(0)
%Initial point for x1,x2 and u
initial_point=0;
%x(1,:)=x1
%x(2,:)=x2
%x(3,:)=u
x0 = initial_point*ones(3,N);
%vector of samples 0..N-1
for i=1:N
    sample(i)=(i-1);
end
%optimization options
options = optimoptions('fmincon','SpecifyObjectiveGradient',false,...
    'Display','iter','StepTolerance',1e-6,'Algorithm','sqp',...
    'MaxIterations',500,'MaxFunctionEvaluations',30000);
%optimization
[x, fval,exitflag,output] = fmincon(@(x) objective_fun(x),x0,[],[],[],[],[],
[],@(x) nonlcon(x), options);

%plots
fig1 = figure('units','normalized','outerposition',[0 0 0.8 0.8]);
subplot(3,1,1);
stairs(sample,x(1,:));
title({'MATLAB - Algorithm SQP',strcat('Plot of x_1(k), J(x,u)=',
num2str(fval))});
```

```
xlabel('k');
ylabel('x_1(k)');
grid on;
subplot(3,1,2);
stairs(sample,x(2,:));
title(strcat('Plot of x_2(k), J(x,u)=', num2str(fval)));
xlabel('k');
ylabel('x_2(k)');
grid on;
subplot(3,1,3);
stairs(sample,x(3,:));
title(strcat('Plot of u(k), J(x,u)=', num2str(fval)));
xlabel('k');
ylabel('u(k)');
grid on;
```

2. AMPL

Pliki programu AMPL umożliwiają rozwiązywanie zadań optymalizacji na własnym komputerze, serwerze NEOS oraz serwerze firmy AMPL.

Wykorzystując serwer NEOS uruchamiamy nasz problem optymalizacji na własnym komputerze jednak zamiast wybierać solwer zainstalowany na własnym komputerze wywołuje się Kestrel, będący programem typu klient, który wysyła zadanie optymalizacji do solwera uruchomionego na zdalnym komputerze serwera NEOS. Na serwerze NEOS występuje znacznie większa liczba solwerów, niż dostępna w podstawowym pakiecie AMPL.

Serwer firmy AMPL umożliwia wprowadzenie plików *.mod oraz *.dat i wybór odpowiedniego solwera do rozwiązania zadania poprzez przeglądarkę internetową. Na serwerze firmy AMPL występuje mniejsza liczba solwerów niż w przypadku serwera NEOS, jednak pozwalają one na rozwiązanie wielu zadań.

Dane zadanie rozwiązano korzystając z solwera MINOS(domyślny) na własnym komputerze oraz z 3 innych solwerów na serwerze AMPL: DONLP2, SNOPT, LOQO. Wszystkie solwery służą do rozwiązywania zadań nieliniowych optymalizacji z ograniczeniami. Tak jak w przypadku programu Matlab optymalizację wykonano startując z różnych punktów.

1. Model – `project2.mod`

```
param t0;                               #start time
param tf;                               #end time
param N;                                #number of samples
param Initial_point;                    #initial point for all var
param x10;                              #x1(0)
param x20;                              #x2(0)
param D = (tf-t0)/N;                    #Delta
set SAMPLES = 0 .. (N-1) by 1;          #set of samples

# decision variables
var x1{SAMPLES}:=Initial_point;
var x2{SAMPLES}:=Initial_point;
var u{SAMPLES}:=Initial_point;

#objective function
minimize Cost_Fun:
      sum{k in SAMPLES} D*( ( x1[k] )^2 + ( x2[k] )^2 + 0.005 * ( u[k] )^2 );

# equality constraint for state x1(0)
```

```
subject to State_X10:
	x1[0] = x10;
# equality constraint for state x2(0)
subject to State_X20:
	x2[0] = x20;
# equality constraints for state x1(k+1)
subject to State_X1_Ceq {k in 0..(N-2)}:
	x1[k+1] - x1[k] - D * x2[k] = 0;
# equality constraints for state x2(k+1)
subject to State_X2_Ceq {k in 0..(N-2)}:
	x2[k+1] - x2[k] - D * (u[k] - x2[k]) = 0;

# nonlinear inequality constraints for state x2(k)
subject to State_X2_C{k in SAMPLES}:
	x2[k] - (8*( D*k - 0.5 )^2 - 0.5) <= 0;
```

2. Dane – `project2.dat`

```
param t0 := 0;                          #start time
param tf := 1;                          #end time
param N := 50;                          #number of samples
param x10 := 0;                         #x1(0)
param x20 :=-1;                         #x2(0)
param Initial_point := -5;              #initial point for all var
```

3. Wykonywalny – `project2.run`

```
reset;
# load model and data files
model project2.mod;
data project2.dat;
option solver minos;
option minos_options 'optimality_tolerance=1e-6';
solve;
# display decision variables
display x1;
display x2;
display u;
```

4. Uzyskane wyniki

W tym rozdziale zaprezentowano wyniki optymalizacji startując z różnych punktów dla wszystkich wykorzystanych solwerów.

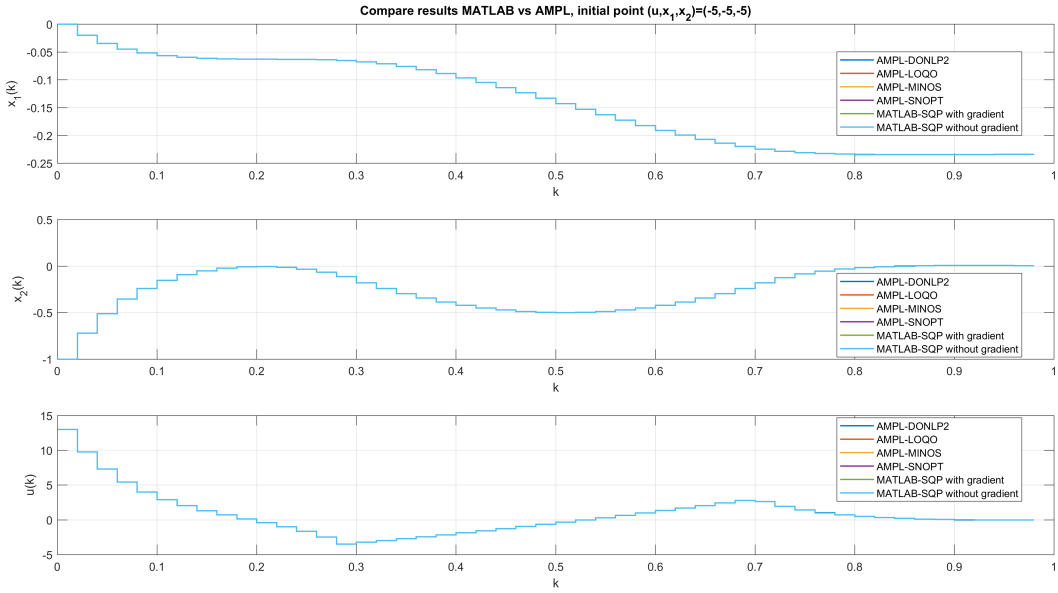
Punkt startowy (x_{1k},x_{2k},u_k) dla k=0..N-1	Wynik optymalizacji					
	Matlab (solwer `fmincon`, algorytm `'sqp'`)		**AMPL**			
	Wersja gradientowa	Wersja bezgradientowa	MINOS	DONLP2	LOQO	SNOPT
(**-5**,**-5**,**-5**)	Wartość funkcji celu: 0.1814 Iteracje: 173	Wartość funkcji celu: 0.1814 Iteracje: 179	Wartość funkcji celu: 0.1814 Iteracje: 136	Wartość funkcji celu: 0.1814 Iteracje: 179	Wartość funkcji celu: 0.1814 Iteracje: 24	Wartość funkcji celu: 0.1814 Iteracje: 146
(**0**,**0**,**0**)	Wartość funkcji celu: 0.1814 Iteracje: 164	Wartość funkcji celu: 0.1814 Iteracje: 163	Wartość funkcji celu: 0.1814 Iteracje: 48	Wartość funkcji celu: 0.1814 Iteracje: 113	Wartość funkcji celu: 0.1814 Iteracje: 22	Wartość funkcji celu: 0.1814 Iteracje: 121
(**5**,**5**,**5**)	Wartość funkcji celu: 0.1814 Iteracje: 178	Wartość funkcji celu: 0.1814 Iteracje: 185	Wartość funkcji celu: 0.1814 Iteracje: 57	Wartość funkcji celu: 0.1814 Iteracje: 238	Wartość funkcji celu: 0.1814 Iteracje: 28	Wartość funkcji celu: 0.1814 Iteracje: 135
(**30**,**30**,**30**)	Wartość funkcji celu: 0.1814 Iteracje: 197	Wartość funkcji celu: 0.1814 Iteracje: 198	Wartość funkcji celu: 0.1814 Iteracje: 64	Wartość funkcji celu: 0.1814 Iteracje: 255	Wartość funkcji celu: 0.1814 Iteracje: 31	Wartość funkcji celu: 0.1814 Iteracje: 142

Tabela 1: Wpływ punktu startowego optymalizacji na otrzymane wyniki

Dla wszystkich solwerów otrzymane wartości funkcji celu w punkcie optymalnym mają taką samą wartość. Zmiana punktu startowego nie wpływa na wartość funkcji celu. Analizując postać funkcji celu można stwierdzić, iż minimalną wartość uzyskałoby się gdyby wartości zmiennych decyzyjnych byłby równe 0. Zatem dla danego punktu startowego równego 0 (dla wszystkich zmiennych decyzyjnych) otrzymano najmniejszą liczbę iteracji algorytmów. W miarę oddalania się od tego punktu startowego liczba iteracji dla każdego solwera zwiększa się. W przypadku analizy działania solwera `fmincon` można stwierdzić, iż w przypadku wersji gradientowej liczba iteracji jest mniejsza.

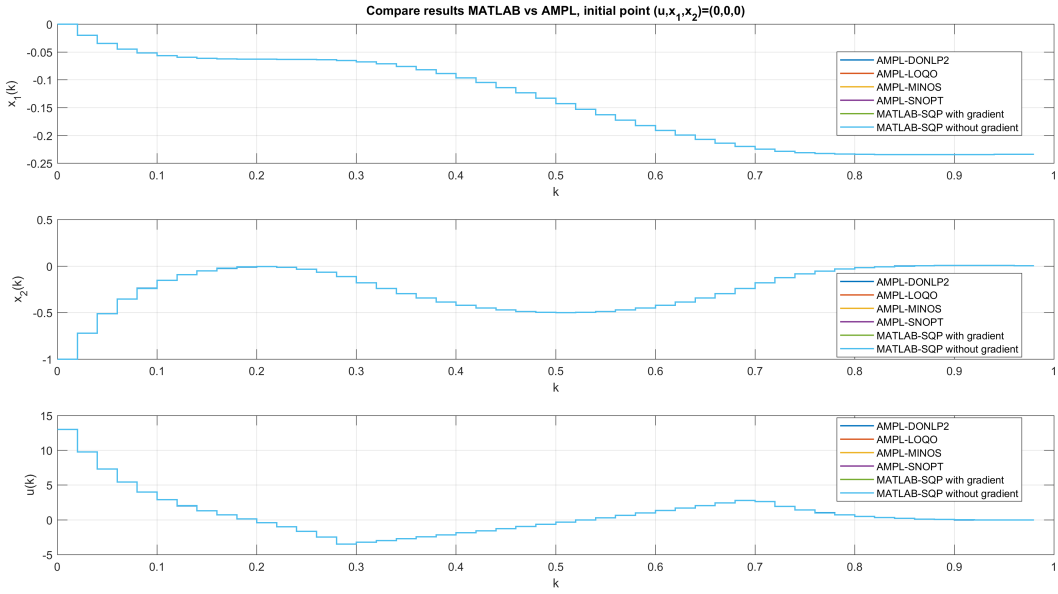
W dalszej części przedstawiono optymalne trajektorie sterowania i stanu(dla różnych punktów startowych) otrzymane wszystkimi solwerami.

- Punkt startowy $(x_{1k},x_{2k},u_k)=(-5,-5,-5)$



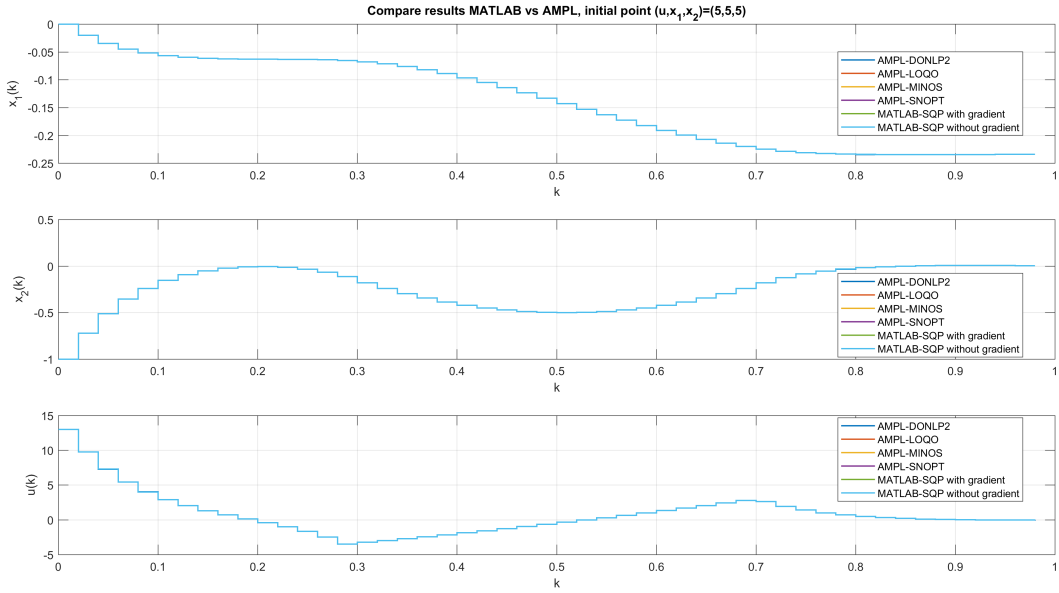
Rysunek 1: Optymalne trajektorie sterowania i stanu dla punktu startowego równego -5 dla wszystkich zmiennych decyzyjnych.

- Punkt startowy $(x_{1k},x_{2k},u_k)=(0,0,0)$



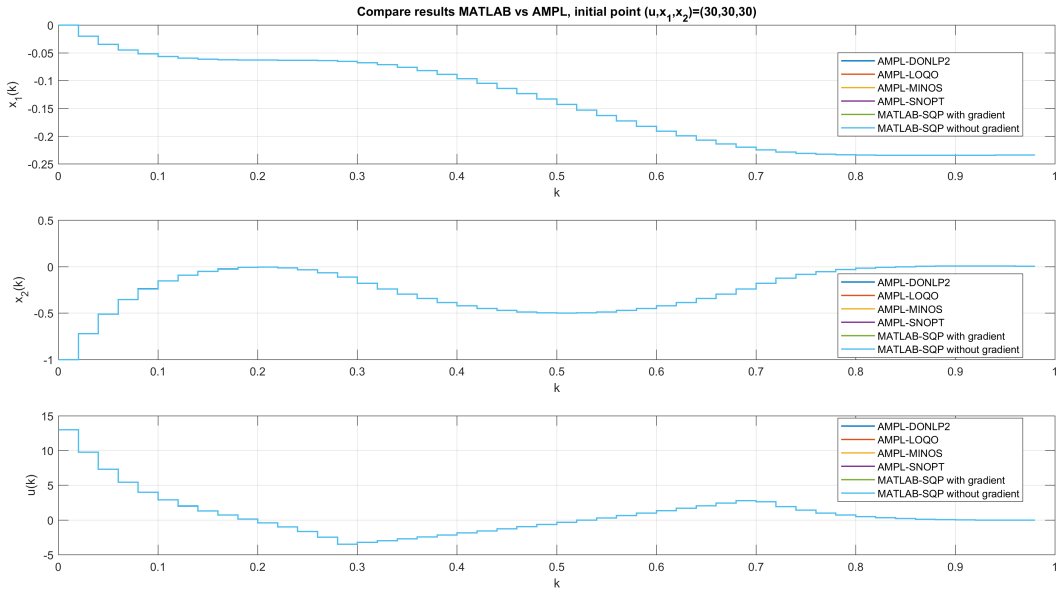
Rysunek 2: Optymalne trajektorie sterowania i stanu dla punktu startowego równego 0 dla wszystkich zmiennych decyzyjnych.

- Punkt startowy $(x_{1k},x_{2k},u_k)=(5,5,5)$



Rysunek 3: Optymalne trajektorie sterowania i stanu dla punktu startowego równego 5 dla wszystkich zmiennych decyzyjnych.

- Punkt startowy $(x_{1k},x_{2k},u_k)=(30,30,30)$



Rysunek 4: Optymalne trajektorie sterowania i stanu dla punktu startowego równego 30 dla wszystkich zmiennych decyzyjnych.

Zarówno otrzymane wyniki jak i trajektorie sterowania i stanu są takie same dla wszystkich solwerów niezależnie od przyjętego punktu startowego. Tak jak wcześniej wspomniano może to wynikać ze ścisłych ograniczeń nałożonych na zmienne decyzyjne wynikające z równań stanu.