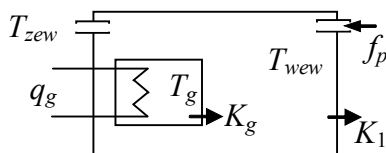


# I. Konstrukcja modeli dynamiki i podstawowe badania symulacyjne

## 1. Konstrukcja modelu dynamiki

### 1.1. Przedmiot badań

Konstrukcja i badania modeli dynamiki zostaną opisane na podstawie przykładowego obiektu cieplnego. Przedmiotem badań będzie więc pomieszczenie z grzejnikiem elektrycznym, które traci ciepło przez ściany (przenikanie ciepła przez przegrody zewnętrzne) oraz w wyniku działania wentylacji (wymiany powietrza w pomieszczeniu).



Rys. I-1. Badany obiekt

Na schemacie badanego obiektu zaznaczono założone kierunki przepływu powietrza i kierunki przekazywania ciepła przez przegrody w warunkach obliczeniowych (od wyższej temperatury do niższej).

Zakładamy, że w typowych warunkach obliczeniowych (to znaczy gdy temperatura na zewnątrz  $T_{zewN}$  wynosi  $-20^{\circ}\text{C}$ ) temperatura wewnątrz pomieszczenia  $T_{wewN}$  wynosi  $20^{\circ}\text{C}$ , grzałka pracuje z mocą  $q_{gN}=1.5\text{kW}$ , a grzejnik osiąga temperaturę  $T_{gN}=45^{\circ}\text{C}$ . Pomieszczenie ma objętość  $V_w=2.5 \cdot 10\text{m}^3$ , a grzejnik  $V_g=10$  litrów.

### 1.2. Opis obiektu cieplnego i planowanych badań

Opis dynamiki obiektu powstaje na bazie chwilowego bilansu ciepła w każdym istotnym magazynie ciepła. Są to równania różniczkowe, powstające według zasady: zmiana ilości ciepła zgromadzonego w danym magazynie ciepła jest równa różnicy pomiędzy strumieniami ciepła wpływającego i wypływającego z tego magazynu. Założymy, że opis obiektu zastosowany w analizowanym przypadku będzie uwzględniać jedynie pojemność cieplną powietrza w pomieszczeniu  $C_{vw}$  i oleju wypełniającego grzejnik  $C_{vg}$ . Model dynamiki obiektu będzie więc zawierał dwa **równania bilansowe (równania dynamiki)** postaci:

$$\begin{cases} C_{vw} \dot{T}_{wew}(t) = K_g(T_g(t) - T_{wew}(t)) - K_1(T_{wew}(t) - T_{zew}(t)) + c_{pp} \rho_p f_p(t) T_{zew}(t) - c_{pp} \rho_p f_p(t) T_{wew}(t) \\ C_{vg} \dot{T}_g(t) = q_g(t) - K_g(T_g(t) - T_{wew}(t)) \end{cases} \quad (\text{I-1})$$

gdzie:

- $C_{vg} T_g$  – akumulacja ciepła w grzejniku wypełnionym olejem,  $C_{vg} = c_{po} \rho_o V_g$ ,
- $C_{vw} T_{wew}$  – akumulacja ciepła w pomieszczeniu (w powietrzu),  $C_{vw} = c_{pp} \rho_p V_w$ ,
- $q_g$  – moc grzałki (strumień ciepła dostarczany do grzejnika),
- $K_g(T_g - T_{wew})$  – strumień ciepła przekazywany przez ściany grzejnika do pomieszczenia,
- $K_1(T_{wew} - T_{zew})$  – strumień ciepła tracony przez ściany pomieszczenia,
- $c_{pp} \rho_p f_p T_{zew}$  – strumień ciepła dostarczany przez powietrze wpływające do pomieszczenia,
- $c_{pp} \rho_p f_p T_{wew}$  – strumień ciepła tracony wraz powietrzem wpływającym z pomieszczenia.

Modele dynamiki będą porządkowane w ten sposób, by różnice temperatur występujące w nawiasach były dodatnie:

$$\begin{cases} C_{vw} \dot{T}_{wew}(t) = K_g(T_g(t) - T_{wew}(t)) - K_1(T_{wew}(t) - T_{zew}(t)) - c_{pp} \rho_p f_p(t)(T_{wew}(t) - T_{zew}(t)) \\ C_{vg} \dot{T}_g(t) = q_g(t) - K_g(T_g(t) - T_{wew}(t)) \end{cases} \quad (\text{I-2})$$

Na podstawie równań bilansowych najłatwiej jest określić **zmienne wejściowe** i **wyjściowe** oraz **parametry** modelu. Wstępne założenie jest takie, że zmiennymi wyjściowymi są te wielkości, które występują w bilansach jako funkcje pochodne (zmienne stanu). Następnie sprawdzamy, czy wszystkie pozostałe wielkości zmienne w czasie są zmiennymi wejściowymi, to znaczy, czy ich stan (wartość) jest wymuszany (określany) na zewnątrz układu (nie zależy od tego co się dzieje na obiekcie). Gdyby tak nie było, to znaczy, że zmiennych wyjściowych jest więcej niż równań różniczkowych i model nie jest kompletny (konieczne są dodatkowe równania). W analizowanym przypadku (I-2) mamy:

- zmienne wyjściowe (zmienne stanu) to  $T_g, T_{wew}$ ,
- pozostałe zmienne ( $q_g, T_{zew}, f_p$ ) są zmiennymi wejściowymi (są wymuszane na zewnątrz układu).

To oznacza, że model jest kompletny. Wszystkie inne wielkości ( $c_{pp}, \rho_p, C_{vg}, C_{vw}, K_1, K_g$ ) są parametrami modelu. Wartości parametrów wpływają na własności modelu i muszą być wyznaczone (zidentyfikowane) dla konkretnego obiektu (p. 1.3.1).

Na podstawie równań dynamiki można wyznaczyć **równania statyczne**, które opisują **stan równowagi (stan ustalony)**, czyli sytuację gdy zmienne wejściowe mają stałą wartość i na obiekcie nie zachodzą żadne zmiany (co oznacza, że wszystkie pochodne mają wartość 0):

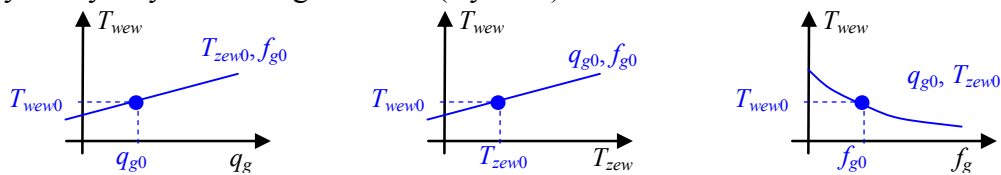
$$\begin{cases} 0 = K_g (T_g - T_{zew}) - K_1 (T_{zew} - T_{zew}) - c_{pp} \rho_p f_p (T_{zew} - T_{zew}) \\ 0 = q_g - K_g (T_g - T_{zew}) \end{cases} \quad (I-3)$$

Ponieważ zmienne mają stałe wartości, nie ma potrzeby oznaczania ich jako funkcji czasu.

Układ równań statycznych można rozwiązać ze względu na zmienne wyjściowe i wyznaczyć **punkt (punkty) równowagi** przy określonych wymuszeniach (wartościach zmiennych wejściowych). Układ równań statycznych (I-3) ma jedno rozwiązanie:

$$T_{zew} = \frac{q_g}{K_1 + c_{pp} \rho_p f_p} + T_{zew}, \quad T_g = \frac{q_g}{K_g} + T_{zew} = \frac{q_g}{K_g} + \frac{q_g}{K_1 + c_{pp} \rho_p f_p} + T_{zew} \quad (I-4)$$

Dla określonych wartości wejściowych ( $T_{zew0}, q_{g0}, f_{p0}$ ) można wyznaczyć punkt równowagi opisany przez dwie zmienne wyjściowe ( $T_{zew0}, T_{p0}$ ). Rozwiązanie (I-4) można wykorzystać do wykreślenia charakterystyk statycznych badanego obiektu (Rys. I-2).



Rys. I-2. Charakterystyki statyczne dla  $T_{zew}$  oraz wybrany punkt równowagi

W przypadku gdy równania statyczne są liniowe to można je zapisać w **postaci macierzowej**:

$$0 = \mathbf{Ax} + \mathbf{Bu} \quad (I-5)$$

gdzie:  $\mathbf{x}$  – wektor zmiennych wyjściowych,  $\mathbf{u}$  – wektor zmiennych wejściowych,  $\mathbf{A}$  i  $\mathbf{B}$  – macierze współczynników. A następnie rozwiązać ze względu na zmienne wyjściowe:

$$\mathbf{x} = -\mathbf{A}^{-1} \mathbf{Bu} \quad (I-6)$$

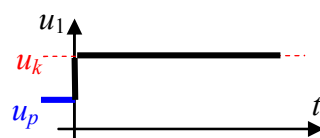
Równania statyczne (I-3) nie są liniowe. Jeśli jednak założyć, że zmienna  $f_p$  jest parametrem ( $f_p = f_{p0}$ ), to można zapisać równania statyczne w postaci macierzowej:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -(K_g + K_1 + c_{pp} \rho_p f_{p0}) & K_g \\ K_g & -K_g \end{bmatrix} \begin{bmatrix} T_{zew} \\ T_g \end{bmatrix} + \begin{bmatrix} 0 & K_1 + c_{pp} \rho_p f_{p0} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_g \\ T_{zew} \end{bmatrix} \quad (I-7)$$

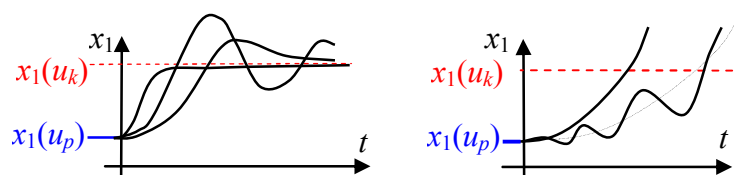
$$[\mathbf{0}] = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u}$$

i stosować rozwiązanie macierzowe (I-6) do obliczania punktu równowagi

Jeśli układ dynamiki zostanie ustawiony dokładnie w punkcie równowagi (punkt równowagi będzie warunkiem początkowym symulacji), to będzie w nim trwał dopóki nie zmienią się wymuszenia. Fakt ten będzie wykorzystywany do weryfikacji poprawności obliczeń i modelu wprowadzonego do programu symulacyjnego (p.2). To, że układ ma punkt równowagi, nie oznacza, że jest to układ stabilny – o tym decyduje pełny opis dynamiki obiektu. Jeśli symulacja zostanie uruchomiona od warunków początkowych różnych od punktu równowagi, to **układ stabilny** osiągnie punkt równowagi, a **układ niestabilny** nie osiągnie punktu równowagi (mimo, że taki punkt istnieje – można go obliczyć). Badanie stabilności oraz wyznaczanie przebiegu dochodzenia układu do stanu równowagi po zmianie wymuszenia należą do podstawowych badań dynamiki obiektów. Najczęściej wykonywane i najprostsze badanie polega na uruchomieniu symulacji od wybranego stanu równowagi (**punktu pracy**) i zarejestrowaniu **reakcji na wymuszenie skokowe** na jednym z wejść (Rys. I-3). Typ i szybkość reakcji pozwala opisać własności dynamiczne obiektu, w tym jego stabilność lub niestabilność (Rys. I-4).



Rys. I-3. Wymuszenie skokowe



Rys. I-4. Typy reakcji układów stabilnych i niestabilnych na wymuszenie skokowe

Sposoby przygotowania i przeprowadzenia takich badań zostaną opisane w kolejnym rozdziale (p.2).

### 1.3. Identyfikacja wartości parametrów obiektu cieplnego

#### 1.3.1. Najprostsze metody szacowania wartości parametrów

Wyznaczanie wartości parametrów musi uwzględniać zgodność wymiarów (jednostek). Zakłada się, że podczas obliczeń i symulacji będą stosowane podstawowe jednostki wielkości, czyli jednostką bilansów energii (strumienia ciepła, mocy) jest zawsze wat [W], a jednostką czasu - sekunda [s]. Stąd współczynniki przenikania ciepła przez poszczególne przegrody ( $K_g$  - ściany grzejnika,  $K_1$  - ściany zewnętrzne) mają wymiar W/K, natomiast pojemności cieplne ( $C_{wv}$ ,  $C_{vg}$ ) – J/K. Przepływ medium (np. powietrza  $f_p$ ) występujący w równaniach bilansowych jest przepływem objętościowym o wymiarze  $m^3/s$ . Ciepło właściwe materiałów ( $c_p$ ) ma wymiar J/(kg·K), a gęstość ( $\rho$ )  $kg/m^3$ . Ponieważ w równaniach bilansowych zawsze występuje różnica temperatur, więc nie ma potrzeby przeliczania temperatur na °K - można stosować °C.

Parametry obiektów cieplnych można podzielić ze względu na sposób wyznaczania (identyfikowania) ich wartości na następujące typy:

- własności materiałów, w szczególności ciepło właściwe ( $c_p$ ) i gęstość ( $\rho$ ), można łatwo znaleźć w różnych zestawieniach materiałów (patrz Załącznik A.1.1),
- wymiary geometryczne – wymiary pomieszczeń, ścian, urządzeń grzewczych, konieczne np. do wyznaczenia objętości magazynów ciepła (w przykładzie  $V_g$ ,  $V_w$ ), są proste do oszacowania,
- parametry „statyczne” - inne parametry, które występują w równaniach statycznych (w przykładzie  $K_1$ ,  $K_g$ ) są wyznaczane przez projektantów tych obiektów na podstawie szczegółowych obliczeń uwzględniających materiał, wymiary geometryczne i konstrukcję (np. kształt, warstwy, ...),
- parametry „dynamiczne”, czyli parametry, które występują tylko w równaniach dynamiki (współczynniki przy zmiennych stanu) – w modelach obiektów cieplnych są to pojemności cieplne magazynów ciepła, które można wyznaczyć na podstawie objętości ( $V$ ) i własności materiału ( $c_p$ ,  $\rho$ ) akumulującego ciepło:

$$C_v = c_p \rho V \quad (I-8)$$

W przybliżonych modelach dynamiki obiektów cieplnych nie potrzeby odtwarzania złożonych obliczeń projektowych - w szczególności dotyczy to wyznaczenia parametrów statycznych, takich jak współczynniki przenikania ciepła przez przegrody. Można je obliczyć na podstawie równań statycznych i wartości zmiennych w warunkach obliczeniowych (nominalnych), które zazwyczaj można łatwo znaleźć w dokumentacji projektowej obiektu lub zmierzyć.

W analizowanym przykładzie układ równań statycznych (I-3) w warunkach obliczeniowych (nominalnych) ma postać:

$$\begin{cases} 0 = q_{gN} - K_g(T_{gN} - T_{wewN}) \\ 0 = K_g(T_{gN} - T_{wewN}) - K_1(T_{wewN} - T_{zewN}) - c_{pp} \rho_p f_{pN} (T_{wewN} - T_{zewN}) \end{cases} \quad (I-9)$$

i znane są następujące wartości:  $T_{zewN} = -20^\circ\text{C}$ ,  $T_{wewN} = 20^\circ\text{C}$ ,  $T_{gN} = 45^\circ\text{C}$ ,  $q_{gN} = 1\,500\text{W}$ .

Nieznany jest natomiast przepływ  $f_{pN}$  oraz współczynnik przenikania ciepła  $K_1$  i  $K_g$ . Ponieważ są dwa równania i trzy niewiadome, to do rozwiązania układu konieczne są jakieś dodatkowe informacje, które można uzyskać lub założyć (oszacować) na temat obiektu. Poniżej przedstawiono kilka możliwych wariantów założeń, które pozwalają obliczyć jedną z trzech niewiadomych:

a) Cztery razy na dobę następuje całkowita wymiana powietrza w pomieszczeniu.

Wymiana cztery razy na dobę, oznacza że w ciągu 6 godzin powietrze o objętości  $V_w$  wypłynie z zewnątrz a wypłynie powietrze z pomieszczenia a, więc:  $f_{pN} = V_w / (6 \cdot 60 \cdot 60)$  [ $m^3/s$ ].

b) W warunkach nominalnych 1/3 ciepła dostarczanego przez grzejnik jest zużywane na wentylację.

Ciepło (strumień ciepła) dostarczany przez grzejnik to  $K_g(T_{gN} - T_{wewN}) = q_{gN}$ , a ciepło (strumień ciepła) zużywane na wentylację to  $c_{pp} \rho_p f_{pN} (T_{wewN} - T_{zewN})$ , tzn. że do układu równań statycznych dochodzi trzecie równanie:  $c_{pp} \rho_p f_{pN} (T_{wewN} - T_{zewN}) = q_{gN} / 3$ .

c) W warunkach nominalnych dobowe zużycie ciepła (zapotrzebowanie) na wentylację to 18kWh.

Jeśli zużycie jest równomierne, to można obliczyć strumień ciepła zużywanego przez wentylację  $q_{wN} = 18000/24 = 750\text{W}$ , stąd trzecie równanie układu to  $c_{pp} \rho_p f_{pN} (T_{wewN} - T_{zewN}) = q_{wN}$ .

d) Współczynnik przenikania ciepła przez ściany jest o rząd mniejszy niż współczynnik przewodzenia grzejnika. Stąd bezpośrednio wynika relacja  $K_1 = 0.1 K_g$ , co stanowi trzecie równanie w układzie równań statycznych.

- e) Jeśli przegrody przewodzące ciepło są tego samego typu (np. ściany z tego samego materiału, albo o tej samej powierzchni czy grubości), to można uzyskać dodatkowe informacje na temat układu porównując parametry, które wpływają na współczynniki przenikania. Na potrzeby badań będziemy przyjmować, że współczynnik przenikania ciepła przez przegrodę (ścianę) można wyznaczyć na podstawie powierzchni przegrody ( $A_w$ , m<sup>2</sup>), grubości ( $a_g$ , m) i jednostkowego współczynnika przenikalności, który zależy od materiału przegrody<sup>1</sup> ( $k$ , W/(mK)):

$$K = k \frac{A_w}{a_g} \quad (I-10)$$

Na podstawie układu (I-9) oraz jednego z założeń (a÷e) można jednoznacznie wyznaczyć wartości  $f_{pN}$ ,  $K_1$  i  $K_g$ . Po wyznaczeniu wartości wszystkich parametrów warto zweryfikować poprawność obliczeń, w szczególności:

- sprawdzić czy wartości współczynników przenikania są dodatnie,
- podstawić wartości parametrów do równania statycznego w warunkach nominalnych (I-9) i sprawdzić zgodność w warunkach obliczeniowych,
- obliczyć punkt równowagi (I-4) dla nominalnych wartości zmiennych wyjściowych i sprawdzić czy rozwiązaniem są nominalne wartości zmiennych wyjściowych.

Weryfikacja poprawności przeprowadzanych operacji jest istotnym elementem metodologii prowadzenia badań symulacyjnych (p.2).

### 1.3.2. Weryfikacja poprawności modelu

Weryfikacja poprawności obliczeń (p. 1.3.1) nie gwarantuje jeszcze poprawności badań symulacyjnych rzeczywistych obiektów. Konieczne jest zweryfikowanie wszystkich założeń, które decydowały zarówno o formie modelu, jak i o wartościach parametrów. Ostatecznie najlepszą weryfikacją modelu jest porównanie reakcji modelu z przebiegami zarejestrowanymi na rzeczywistym obiekcie. [Opisać metodę](#)

Jeśli dane z rzeczywistego obiektu nie są dostępne, można próbować wyznaczać i porównywać inne parametry (wskaźniki) stosowane w charakterystykach rzeczywistych obiektów. W przypadku budynków są dostępne na przykład klasyfikacje budynków pod względem zapotrzebowania na ciepło czy projekty budowlane konkretnych obiektów (patrz Załącznik A.1.2). W przypadku różnego typu grzejników (wymienników ciepła) trudniej o podobne wskaźniki, ponieważ wartości projektowe takie jak wydajność (moc) wymiennika zależy nie tylko od wielkości (powierzchni), ale także kształtu i sposobu montażu.

Wskaźniki przytoczone powyżej, pozwalają zweryfikować model obiektu w warunkach statycznych. Weryfikacja poprawności modelu dynamiki musi jeszcze uwzględniać zagadnienia decydujące o własnościach dynamicznych, na przykład czy model uwzględnia najbardziej istotne magazyny ciepła? Najistotniejsze są te magazyny, które mają dużą pojemność cieplną, a to zależy nie tylko od objętości magazynu, ale także od materiału. W analizowanym przykładzie modelu (I-2) uwzględniono tylko pojemność cieplną powietrza i grzejnika, a nie uwzględniono pojemności cieplnej ścian, która najprawdopodobniej jest najistotniejsza w tego typu obiektach (porównaj z p. 7.3.1).

Sposób konstrukcji i obliczania (szacowania) wartości parametrów obiektów cieplnych, opisany w p.1.3.1 i zastosowany w symulacji w p.2, nie uwzględnia korelacji pomiędzy parametrami statycznymi (współczynnikami przenikania ciepła  $K_1$ ,  $K_g$ ) i dynamicznymi (pojemnościami cieplnymi magazynów  $C_{vw}$ ,  $C_{vg}$ ). Parametry te były wyznaczone niezależnie, co jest znacznie prostsze i wystarczające w przybliżonych modelach obiektów cieplnych. Przy dokładniejszym odwzorowaniu rzeczywistego obiektu należałoby uwzględnić, że oba typy parametrów zależą od wymiarów geometrycznych i własności materiałów. Przecież można zbudować dwa budynki o tej samej kubaturze i zużyciu ciepła, na przykład jeden drewniany a drugi z cegły, a wówczas:

- współczynniki przenikania ciepła przez ściany budynków są takie same,
- budynek drewniany będzie miał znacznie mniejszą pojemność cieplną, ponieważ drewno ma gorszy współczynnik przewodności  $\lambda$  (wystarczy cieńsza ściana) i jest lżejsze od cegły.

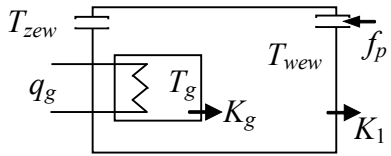
Przykładowe obliczenia takich relacji przedstawiono w Załączniku A.2).

<sup>1</sup> Zastosowanie jednostkowego współczynnika  $k$  przenikania jest uproszczeniem, które nie uwzględnia, że przenikanie ciepła zależy nie tylko od wymiarów i materiału przegrody, ale także od warunków w otoczeniu przegrody (p.Zał. A.2.1)

#### 1.4. Podsumowanie opisu obiektu

W Tab. I-1 zebrano wszystkie wzory i dane potrzebne do wprowadzenia modelu ( I-1) w programie symulacyjnym.

Tab. I-1. Zbiorczy opis modelu ( I-1)

 <p><math>C_{vw} = c_{pp} \rho_p V_w</math>, <math>C_{vg} = c_{po} \rho_o V_g</math></p>	<p>Materiały:</p> <ul style="list-style-type: none"> <li>- powietrze: <math>c_{pp}=1000 \text{ J/(kgK)}</math>; <math>\rho_p=1.2 \text{ kg/m}^3</math></li> <li>- olej: <math>c_{po}=2400 \text{ J/(kgK)}</math>; <math>\rho_o=1200 \text{ kg/m}^3</math></li> </ul> <p>Objętości: <math>V_w = 2.5 \cdot 10^3 \text{ m}^3</math>, <math>V_g = 10/1000 \text{ m}^3</math></p> <p>Wartości obliczeniowe (nominalne) + założenia a:</p> <p><math>T_{zewN} = -20^\circ\text{C}</math>, <math>T_{wewN} = 20^\circ\text{C}</math>, <math>T_{gN} = 45^\circ\text{C}</math>, <math>q_{gN} = 1\,500 \text{ W}</math></p> <p><math>f_{pN} = V_w / (6 \cdot 60 \cdot 60) \text{ m}^3/\text{s}</math></p>
<p>Model dynamiki:</p> $\begin{cases} C_{vw} \dot{T}_{wew}(t) = K_g (T_g(t) - T_{wew}(t)) - K_1 (T_{wew}(t) - T_{zew}(t)) - c_{pp} \rho_p f_p(t) (T_{wew}(t) - T_{zew}(t)) \\ C_{vg} \dot{T}_g(t) = q_g(t) - K_g (T_g(t) - T_{wew}(t)) \end{cases}$	
<p>Model dynamiki uproszczony („zlinearyzowany”)</p> $\begin{cases} C_{vw} \dot{T}_{wew}(t) = K_g (T_g(t) - T_{wew}(t)) - K_1 (T_{wew}(t) - T_{zew}(t)) - c_{pp} \rho_p f_{p0} (T_{wew}(t) - T_{zew}(t)) \\ C_{vg} \dot{T}_g(t) = q_g(t) - K_g (T_g(t) - T_{wew}(t)) \end{cases}$	
<p>Stan ustalony:</p>	$\begin{cases} 0 = K_g (T_g - T_{wew}) - K_1 (T_{wew} - T_{zew}) - c_{pp} \rho_p f_p (T_{wew} - T_{zew}) \\ 0 = q_g - K_g (T_g - T_{wew}) \end{cases}$
<p>Identyfikacja (wariant a):</p>	$f_{pN} = V_w / (6 \cdot 60 \cdot 60)$ $K_g = \frac{q_{g0}}{T_{gN} - T_{wewN}}, \quad K_1 = \frac{q_{gN} - c_{pp} \rho_p f_p (T_{wew} - T_{zew})}{T_{wewN} - T_{zewN}}$
<p>Punkt równowagi:</p>	$T_{wew0} = \frac{q_{g0}}{K_1 + c_{pp} \rho_p f_{p0}} + T_{zew0}$ $T_g = \frac{q_g}{K_g} + T_{wew} = \frac{q_g}{K_g} + \frac{q_g}{K_1 + c_{pp} \rho_p f_p} + T_{zew}$

Klasyfikacja: model nieliniowy drugiego rzędu

Zmienne wejściowe:  $q_g, T_{zew}, f_p$

Zmienne wyjściowe:  $T_{wew}, T_g$

## 2. Konstrukcja, weryfikacja i podstawowe badania modeli w trybie wsadowym

### 2.1. Cel i założenia badań symulacyjnych

Badania symulacyjne z wykorzystaniem takich programów jak Matlab i Scilab, można prowadzić na różne sposoby, na przykład w zależności od:

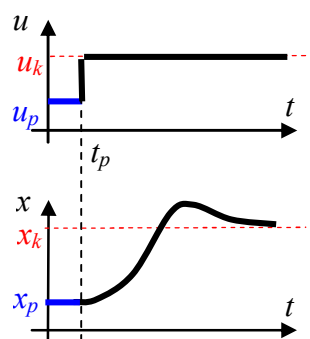
- metody przygotowania modelu - w trybie graficznym (schematy równań różniczkowych) lub w trybie tekstowym (równania różniczkowe zapisane za pomocą odpowiednich funkcji),
- techniki uruchamiania obliczeń – on-line (kolejne polecenia są wpisywane lub wybierane z menu) lub wsadowo (przygotowanie i uruchamianie symulacji za pomocą skryptu),
- stopnia uogólnienia rozwiązania – operacje na wartościach lub zastosowanie symboli (parametrów).

Podstawowym założeniem rozwiązań przedstawionych poniżej jest, aby cały program badań był zapisany w skrypcie, który zapewni zautomatyzowanie i dokumentowanie badań oraz ułatwi edycję założeń i powtórzenie badań. Skrypty będą realizować następujące zadania:

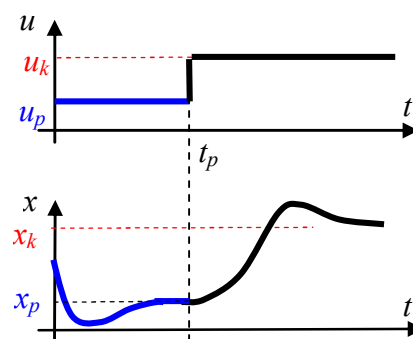
- zdefiniowanie danych wejściowych i obliczanie wartości parametrów modelu ze wzorów,
- przygotowanie i uruchomienie symulacji według założonego programu badań,
- wygenerowanie wykresów, które będą podstawowym sposobem prezentacji wyników.

Przykłady skryptów w punktach 2.2÷2.4 (por.Zał.B.1) realizują program badań ograniczony do wygenerowania reakcji na jedno wybrane zakłócenie, w jednym wybranym punkcie pracy. Przykłady można łatwo rozbudować, tak by służyły do przeprowadzenia prostej analizy czasowej - jak obiekt reaguje na takie samo zakłócenie w różnych punktach pracy.

Badanie reakcji obiektu na skokową zmianę wartości wybranego wejścia, w różnych punktach pracy jest najprostszym badaniem własności dynamicznych obiektu i jest również często wykonywane w warunkach doświadczalnych na rzeczywistych obiektach. Poprawna realizacja tego badania wymaga, aby **skok wartości na wejściu następował zawsze w warunkach stanu ustalonego**. Symulacje będą więc być uruchamiane od odpowiedniego stanu ustalonego (punktu równowagi, nazywanego również punktem pracy). Dla zapewnienia kontroli tego warunku, wystąpienie skoku będzie przesunięte w czasie ( $t_p$ ), tak aby na wykresach był widoczny początkowy stan równowagi (Rys. I-5), to znaczy dopóki nie ma zmian na wejściu ( $u$ ), wszystkie wartości wyjściowe ( $x$ ) są stałe.



Rys. I-5. Badanie reakcji na wymuszenie skokowe (1)



Rys. I-6. Badanie reakcji na wymuszenie skokowe (2)

Możliwe jest też alternatywne rozwiązanie poprawnej realizacji badania w przypadkach gdy symulacja nie jest uruchamiania od stanu równowagi (Rys. I-6) – skok wartości wejściowej powinien nastąpić dopiero w momencie ( $t_p$ ), gdy układ osiągnie początkowy stan równowagi. Opowiada to warunkom przeprowadzania eksperymentu na rzeczywistych obiektach i jest możliwe tylko dla układów stabilnych. Badanie reakcji na skok wartości wejściowej w momencie, gdy układ nie jest w stanie równowagi jest bezużyteczne z punktu widzenia celowości podstawowych badań dynamiki, podobnie jak badanie reakcji na zmiany wielu wartości wejściowych jednocześnie. Ponieważ właściwy eksperyment rozpoczyna się w momencie wystąpienia skoku, to podczas prezentacji wyników, a także w opracowaniach teoretycznych, przedstawia się zazwyczaj reakcję na wymuszenie skokowe występujące w chwili  $t_p=0$  (Rys. I-3, Rys. I-4), co jednak zawsze oznacza gwarancję początkowego stanu równowagi.

Sposoby realizacji badań w programach symulacyjnych, przedstawione w kolejnych punktach, zakładają uogólnienie przygotowywanych schematów i skryptów, przez zastosowanie symboli wartości. Celem takiego podejścia jest przygotowanie plików symulacyjnych w taki sposób, aby zapewnić automatyczne przystosowanie do zmiany założonych wartości obliczeniowych. Tak więc wszystkie obliczenia są realizowane w skrypcie na podstawie wpisanych tam wzorów (identyfikacja wartości parametrów, obliczanie punktu pracy), a schematy są sparаметryzowane (jako parametry bloków nie wpisuje się konkretnych wartości, tylko symbole lub wzory). Programy symulacyjne nie rozróżniają symboli oznaczających zmienne modelu i stałe parametry – wszystko są to zmienne definiowane w przestrzeni roboczej programu. W Tab. I-2 przedstawiono listę zmiennych i parametrów modelu oraz odpowiadające im zmienne programu stosowane w prezentowanych skryptach.

Tab. I-2. Oznaczenia zmiennych w modelu i w programie symulacyjnym.

Oznaczenie we wzorach	Zmienne modelu			Parametry modelu	
	wartość obliczeniowa	wartość początkowa	wynik symulacji	Oznaczenie we wzorach	Oznaczenie w skrypcie
$t$			t	$c_{pp}$ , J/(kgK)	cpp = 1000
$T_{zew}$ , °C	TzewN = -20	Tzew0		$\rho_p$ , kg/m <sup>3</sup>	rp = 1.2
$q_g$ , W	QgN = 1500	Qg0		$c_{po}$ , J/(kgK)	cpo = 2400
$f_p$ , m <sup>3</sup> /s	FpN	Fp0		$\rho_o$ , kg/m <sup>3</sup>	ro = 1200
$T_{wew}$ , °C	TwewN = 20°C	Twew0	aTwew	$V_g$ , m <sup>3</sup>	Vg=0.01
$T_g$ , °C	TgN = 45°C	Tg0	aTg	$V_w$ , m <sup>3</sup>	Vw=100*2.5
				$C_{vg}$ , J/K	Cvg
				$C_{vw}$ , J/K	Cvw
				$K_1$ , W/K	K1
				$K_g$ , W/K	Kg

W skryptach stosowane są poza tym zmienne opisujące parametry wymuszenia skokowego - wartość skoku ( $dT_{zew}$ ,  $dQ_g$ ,  $dF_g$ ) i moment wystąpienia skoku ( $czas\_skok$ ).

## 2.2. Modele obiektów liniowych/nieliniowych „w trybie graficznym” i analiza czasowa

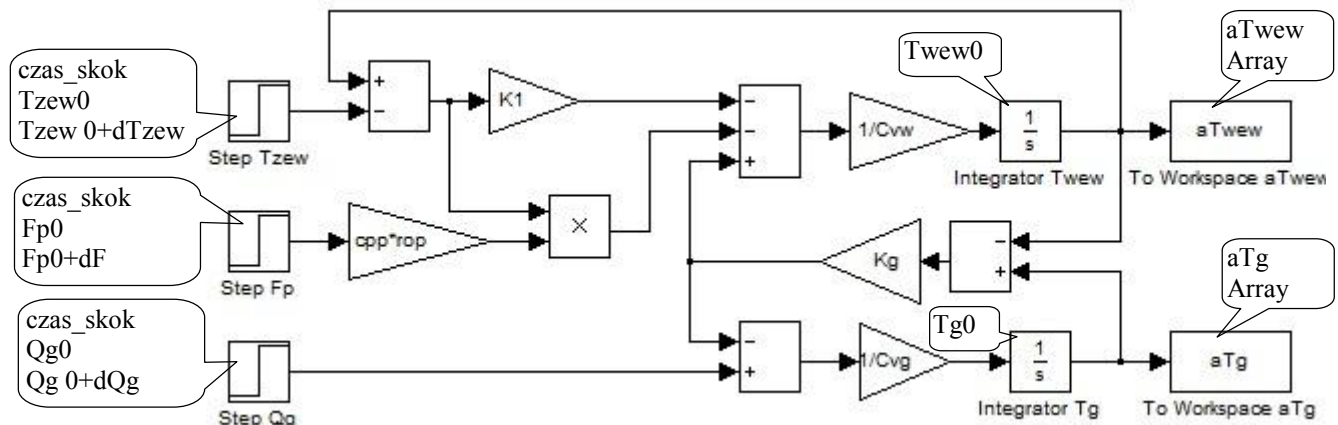
### 2.2.1. Założenia

Przedstawiony sposób realizacji badań polega na narysowaniu schematu modelu dynamiki przy wykorzystaniu interfejsu graficznego, jaki udostępnia zarówno program Matlab z przybornikiem Simulink, jak i program Scilab (funkcja Xcos). Ponieważ badany model jest nieliniowy, więc schemat modelu zostanie wykonany ogólną metodą, to znaczy w oparciu o bloki całkujące (Integrator) [1]. Ze schematem stowarzyszony będzie skrypt, zawierający wszystkie konieczne definicje i obliczenia oraz uruchamianie symulacji i rysowanie wykresów.

Schemat będzie sparametryzowany, to znaczy, że nie będzie zawierać żadnych stałych wartości. Schemat będzie przygotowany w ten sposób, umożliwić badanie reakcji na skokową zmianę wartości na poszczególnych wejściach ( $q_e$ ,  $T_{zew}$ ,  $f_p$ ) w punkcie pracy ( $T_{wew0}$ ,  $T_{gp0}$ ) wyznaczonym na podstawie początkowych wartości wejściowych ( $Q_{g0}$ ,  $T_{zew0}$ ,  $F_{g0}$ ). Wartości zmiennych wyjściowych ( $T_g$ ,  $T_{wew}$ ) będą zapamiętywane jako wektory wartości ( $aT_{wew}$ ,  $aT_g$ ). Przygotowanie i sterowanie symulacjami ma się odbywać za pomocą skryptu (*wsadowo*).

### 2.2.2. Aplikacja modelu w środowisku Matlab/Simulink

Schemat równań dynamiki (I-2) przygotowany według założeń 2.2.1 przedstawiono na Rys. I-7. W dymkach przedstawiono sposób zdefiniowania parametrów w blokach Step (Step time, Initial value, Final value), Integrator (Initial condition) oraz To Workspace (Variable name, Save format). Schemat został zapamiętany w pliku o nazwie *obl.mdl*.



Rys. I-7. Schemat modelu w oparciu o bloki całkujące w środowisku Simulink (*obl.mdl*)

**Skrypt (*obl\_skrypt1.m*)** inicjujący zmienną i uruchamiający symulację modelu *obl.mdl*:

```

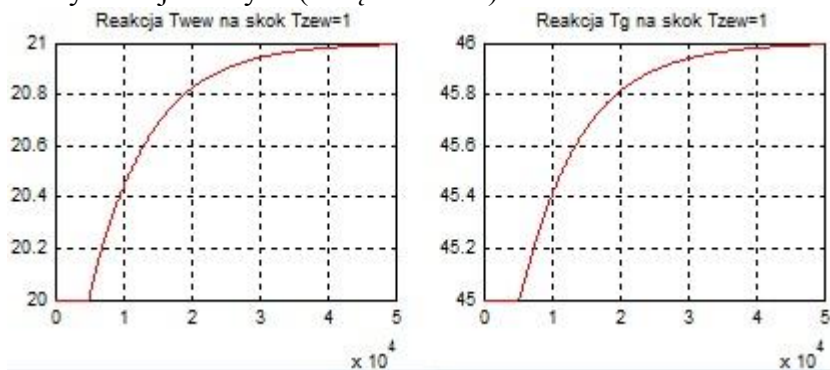
§===== I część (identyfikacja) =====
§wartości nominalne
TzewN=-20;           %oC
TwewN=20;           %oC
TgN=45;             %oC
QgN=1500;           %W
Vg=10/1000;         %m3, objetosc grzenika
Vw=2.5*10*10;       %m3, objetosc pokoju
cpp=1000; rop=1.2;  %J/(kgK), kg/m3, powietrze
cpo=2400; roo=1200; %J/(kgK), kg/m3, olej
§-----
§identyfikacja parametrów statycznych
FpN=Vw/(6*60*60);   %zał.a: wymiana 4x na dobę
Kg=QgN/(TgN-TzewN);
Kl=(QgN-cpp*rop*FpN*(TwewN-TzewN))/(TwewN-TzewN);
§-----
§parametry "dynamiczne"
Cvg=cpo*roo*Vg;
Cvw=cpp*rop*Vw;

§===== II część (punkt pracy) =====
§warunki początkowe
Tzew0= TzewN + 0;   %0, 10, 20
Qg0 = QgN * 1.0;    %1.0, 0.7, 0.3
Fp0 = FpN * 1.0;    %1.0, 0.7, 0.3
§-----
§stan równowagi
Twew0 = Qg0/(Kl+cpp*rop*Fp0)+Tzew0;
Tg0 = Qg0/Kg + Twew0;

§===== III część (symulacje) =====
czas = 50000;       %czas symulacji
§zakłócenie
czas_skok = 5000;   %moment wystąpienia skoku
dTzew=1;           %0 lub np. 1
dQg=0;             %0 lub np. 0.05*QgN (5% zakresu)
dFp=0;             %0 lub np. 0.2*FpN (20% zakresu)
§-----
§symulacja
[t]=sim('obl',czas); %t=wektor czasu
§-----
§wykresy
figure, plot(t,aTwew,'r'), grid on, title('Reakcja Twew na skok Tzew');
figure, plot(t,aTg,'r'), grid on, title('Reakcja Tg na skok Tzew');

```

Uruchomienie skryptu spowoduje wygenerowanie reakcji na skok temperatury  $T_{zew}$  o  $1^{\circ}\text{C}$  przy nominalnych warunkach początkowych (Rys. I-8). Ustalając wartości początkowe zmiennych wejściowych ( $Qg0$ ,  $Tzew0$ ,  $Fp0$ ) można zmienić punkt pracy, natomiast ustalając zerowe lub niezerowe wartości skoków ( $dQg$ ,  $dTzew$ ,  $dFp$ ) można badać reakcje na skokowe zmiany poszczególnych zmiennych wejściowych (Załącznik B.2).

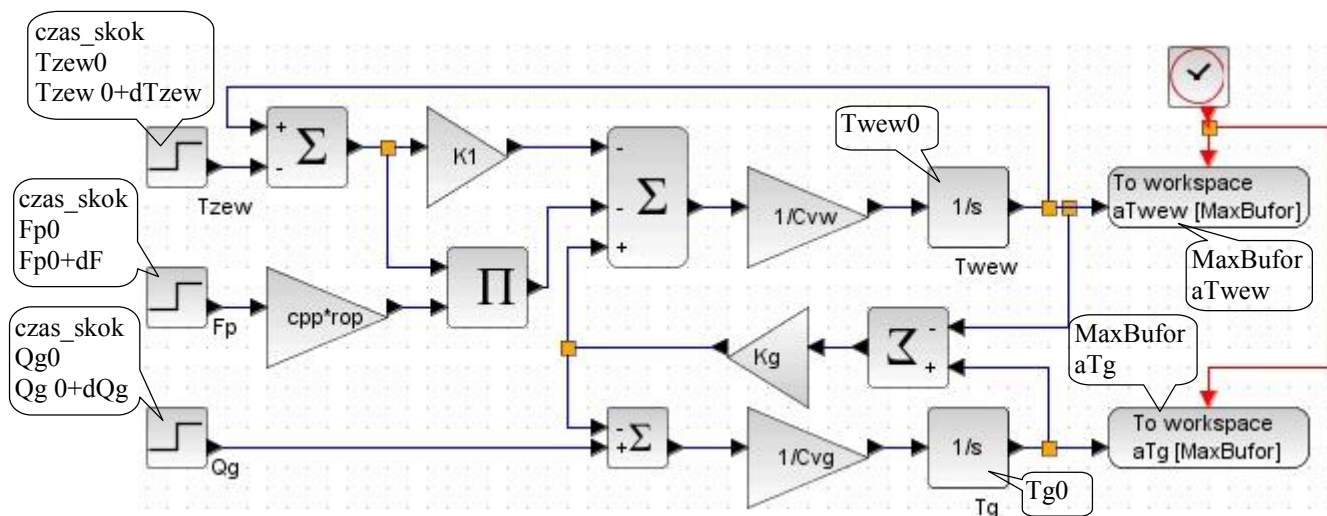


Rys. I-8. Wynik działania skryptu *obl\_skrypt1.m* (wykresy po sformatowaniu – patrz Załącznik B.3)

### 2.2.3. Aplikacja modelu w środowisku Scilab-Xcos

Schemat równań dynamiki (I-2) przygotowany zgodnie z założeniami 2.2.1, zrealizowany w środowisku Xcos, przedstawiono na Rys. I-9. Zastosowano najprostsze bloki z biblioteki Xcos, a w dymkach przedstawiono sposób zdefiniowania parametrów w blokach Step\_Function (Step Time, Initial value, Final value), Integral\_f (Initial state) oraz To Workspace (Size of buffer, Scilab variable name). Schemat został zapamiętany w pliku o nazwie *obl.xcos*.



Rys. I-9. Schemat modelu w oparciu o bloki całkujące w środowisku Xcos (*ob1.mdl*)

**Skrypt (*ob1\_skrypt1.sce*)** inicjujący zmienne i uruchamiający symulację modelu *ob1.xcos*:

```

//===== I czesc (identyfikacja) =====
//wartosci nominalne
TzewN=-20; //oC
TwewN=20; //oC
TgN=45; //oC
QgN=1500; //W
Vg=10/1000; //m3, objetosc grzenika
Vw=2.5*10*10; //m3, objetosc pokoju
cpp=1000; rop=1.2; //J/(kgK), kg/m3, powietrze
cpo=2400; roo=1200; //J/(kgK), kg/m3, olej
//-----
//identyfikacja parametrów statycznych
FpN=Vw/(6*60*60); //zał.a: wymiana 4x na dobę
Kg=QgN/(TgN-TwewN);
K1=(QgN-cpp*rop*FpN*(TwewN-TzewN))/(TwewN-TzewN);
//-----
//parametry "dynamiczne"
Cvg=cpo*roo*Vg;
Cvw=cpp*rop*Vw;
//===== II czesc (punkt pracy) =====
//warunki początkowe
Tzew0= TzewN + 0; //0, 10, 20
Qg0 = QgN * 1.0; //1.0, 0.7, 0.3
Fp0 = FpN * 1.0; //1.0, 0.7, 0.3
//-----
//stan równowagi
Twew0 = Qg0/(K1+cpp*rop*Fp0)+Tzew0;
Tg0 = Qg0/Kg + Twew0;
//===== III czesc (symulacja) =====
loadXcosLibs(); //wczytanie bibliotek xcos
importXcosDiagram('C:\projekty\ob1.xcos'); //wczytanie modelu (tworzy zmienną scs_m)
czas = 50000; zegar=2; //czas symulacji i parametr: Clock/Period
MaxBufor = czas*10; //parametr To workspace/Size of buffer (było czas/zegar)
scs_m.props.tf=czas; //czas trwania symulacji (zamiast czasu ze schematu)
//zakłócenie
czas_skok = 5000; //moment wystąpienia skoku
dTzew=1; //0 lub 1
dQg=0; //0 lub 0.05*QgN
dFp=0; //0 lub 0.2*FpN
//-----
//symulacja i wykresy
scicos_simulate(scs_m); //czas symulacji: scs_m.props.tf
figure , plot(aTwew.time,aTwew.values,'red'), xgrid(2), title('Reakcja Twew na skok Tzew');
figure , plot(aTg.time,aTg.values,'red'), xgrid(2), title('Reakcja Tg na skok Tzew');

```

Uwagi do skryptu: 1) Podając nazwę schematu, należy podać pełną ścieżkę. 2) Blok To Workspace zawsze zapamiętuje wartości w postaci struktury: time, values. Trzeba dobrać wystarczającą wielkość bufora. 3) **Jeśli podczas wykonywania symulacji pojawi się komunikat: „przekroczono rozmiar stosu!”, należy wykonać: stacksize('max').**

#### 2.2.4. Procedura konstrukcji i weryfikacji modelu – uogólnienie na podstawie przykładu

Istotnym elementem prowadzenia badań symulacyjnych jest weryfikacja poprawności wykonywanych operacji na każdym etapie. Praktyka pokazuje, że nawet podczas realizowania prostych badań zdarzają się pomyłki i trudno je zauważyć. Najłatwiej zweryfikować poprawność symulacji, gdy wiadomo jaki ma być wynik, np. można go potwierdzić analitycznie. Tymczasem badania symulacyjne mają największe znaczenie podczas rozwiązywania zadań, których nie można rozwiązać analitycznie i trudno jest przewidzieć rezultat.

Weryfikacja powinna być prowadzona na bieżąco podczas przygotowywania schematu i skryptu. Podstawą proponowanej metody jest sprawdzanie zachowania modelu w stanie równowagi - jeśli symulacja zostanie uruchomiona dokładnie w punkcie równowagi, to niezależnie od stabilności modelu, stan równowagi będzie zachowany dopóki wartości wejściowe nie zmienią się. W prezentowanej procedurze można wyróżnić 4 etapy, które kończą się weryfikacją poprawności zrealizowanych działań - Tab. I-3.

Tab. I-3. Etapy przygotowania i weryfikacji plików symulacyjnych.

Etapy:	Działania	Weryfikacja poprawności
punkty 1÷2	Identyfikacja wartości	Założone wartości nominalne i obliczone wartości współczynników muszą spełniać układ równań statycznych.
punkty 3÷5	Przygotowanie i uruchomienie symulacji w nominalnym punkcie pracy bez zakłóceń	Model zachowuje nominalny stan równowagi (np. stałe wartości wyjściowe) - to oznacza, że prawdopodobnie identyfikacja (obliczenia) wartości parametrów i schemat modelu są poprawne
punkt 6	Przygotowanie wzorów do obliczania punktu pracy	Jeśli wartości wejściowe są nominalne, to wartości wyjściowe obliczone na podstawie wzoru też powinny być nominalne - to potwierdza poprawność wzoru na punkt równowagi.
punkt 7	Uruchomienie symulacji w dowolnym punkcie pracy bez zakłóceń	Model zachowuje dowolny stan równowagi - to wzmacnia prawdopodobieństwo poprawności schemat modelu.
punkt 8	Uruchomienie symulacji w dowolnym punkcie pracy ze skokiem na wejściu	Model zachowuje dowolny stan równowagi do momentu wystąpienia skoku, następnie (o ile jest stabilny) dąży do osiągnięcia nowego stanu równowagi

**Kolejne kroki przygotowywania i stopniowej weryfikacji plików symulacyjnych (schematu i skryptu) są następujące:**

1. Określenie zmiennych wejściowych ( $u$ ) i wyjściowych ( $x$ ) modelu dynamiki (potwierdzenie kompletności modelu).
2. Identyfikacja wartości współczynników (**I część** w pliku skryptu)
  - a) Z równań statycznych modelu, na podstawie podanych wartości nominalnych (obliczeniowych) i dodatkowych założeń wyprowadzić wzory do obliczania współczynników modelu.
  - b) Zainicjować w skrypcie nominalne wartości zmiennych wejściowych i wyjściowych, tzn. zmienne wejściowe  $U_N$  i wyjściowe  $X_N$ , pod które podstawia się podane wartości nominalne.
  - c) Wpisać w skrypcie wzory na współczynniki ( $K$ ), wykorzystując zmienne nominalne.
  - d) Zainicjować pozostałe zmienne (np. parametry opisujące dynamikę obiektu).
  - e) Sprawdzić poprawność obliczeń, na przykład podstawiając wszystkie wartości do równań statycznych. Jeśli równania nie są spełnione, to błąd w obliczeniach lub w założeniach (np. za dużo założeń, co prowadzi do sprzeczności).
3. Ustalenie warunków początkowych w warunkach nominalnych (**II część** w pliku skryptu)
  - a) Zainicjować w skrypcie wartości początkowe zmiennych wejściowych ( $U_0$ ) i nadać im wartości nominalne ( $U_0 = U_N$ ).
  - b) Zainicjować w skrypcie wartości początkowe zmiennych wyjściowych ( $X_0$ ) i nadać im wartości nominalne ( $X_0 = X_N$ ) – nie wprowadzać jeszcze wzorów na stan równowagi.
4. Zdefiniowanie zakłóceń (**III część** w pliku skryptu) – bez zakłóceń
  - a) Zainicjować w skrypcie zmienne opisujące zakłócenia na wejściach, o zerowych wartościach ( $dU_0 = 0$ )
5. Aplikacja modelu w oknie Simulink/Xcos (**schemat**)
  - a) Narysować schemat (jako parametry bloków używać zmienne zdefiniowane w skrypcie).

- b) W „końcowych” blokach całkujących podstawić jako warunki początkowe odpowiednie zmienne  $X_0$  (wartości początkowe pozostałych bloków całkujących pozostają na domyślnej wartości 0).
  - c) Jako źródła wejściowe zastosować bloki skoku, w których jako wartości początkowe zostaną podstawione odpowiednie zmienne  $U_0$ , a jako wartości końcowe – wyrażenia  $U_0 + dU_0$ . Warto wprowadzić zmienną określającą moment wystąpienia skoku (np. wspólna zmienna *czas\_skok* dla wszystkich bloków skoku, o wartości ustawianej w skrypcie).
  - d) Uruchomić symulację - układ powinien być w stanie równowagi, ponieważ zakłócenia są zerowe ( $dU_0 = 0$ ). Jeśli na wyjściach występują stałe wartości, równe wartościom nominalnym, to wskazuje na poprawność aplikacji modelu. Jeśli nie, to jest błąd - prawdopodobnie na schemacie.
6. Obliczanie punktu pracy (warunki początkowe w dowolnym punkcie równowagi)
- a) Z modelu statycznego na podstawie obliczonych współczynników i założonych wartości początkowych zmiennych wejściowych ( $U_0$ ) wyprowadzić wzory na zmienne wyjściowe ( $X_0$ ) w stanie ustalonym.
  - b) Wprowadzić wzory na zmienne wyjściowe ( $X_0$ ) do skryptu (w II części skryptu podmienić podstawienia  $X_0 = X_N$  na wzory).
  - c) Sprawdzić poprawność wzorów i skryptu – jeśli zmienne wejściowe mają wartości nominalne ( $U_0 = U_N$ ), to wartości zmiennych wyjściowych obliczone ze wzorów też powinny być nominalne ( $X_0 = X_N$ ). Jeśli nie, to jest błąd w wyprowadzeniu wzorów lub zapisania ich w skrypcie.
7. Uruchomienie symulacji od dowolnego punktu równowagi
- a) Zmienić w skrypcie wartości początkowe zmiennych wejściowych (II część skryptu), na przykład względem wartości nominalnych:  $U_0 = U_N * 0.5$ , czyli 50% wartości nominalnej; lub  $U_0 = U_N - 1$ , czyli o 1 mniej niż wartość nominalna.
  - b) Uruchomić skrypt i symulację – ponieważ wartości wejściowe są stałe (brak zakłóceń), to na wyjściach też powinny być stałe wartości. Jeśli nie, to zazwyczaj wskazuje na błąd, polegający na wpisaniu wartości nominalnych zamiast początkowych na schemacie lub w skrypcie.
8. Zbadanie reakcji na skokową zmianę wartości wybranych wielkości wejściowych
- a) Ustawić w skrypcie wartość zakłócenia na wybranym wejściu (III część skryptu), podając wprost wartość (np.  $dU_0 = 1$ ) lub względem wartości nominalnych (np.  $dU_0 = U_N * 0.1$ , czyli 10% wartości nominalnej).
  - b) Uruchomić skrypt i symulację – do momentu wystąpienia skoku na wyjściach powinny być widoczne stałe wartości, a od momentu wystąpienia skoku pojawia się zmiana. Jeśli początkowy stan równowagi nie jest widoczny, to przesunąć moment podawania skoku (zmienna *czas\_skok*).

Do weryfikacji poprawności skryptu i schematu można również wykorzystać porównanie wyników otrzymanych przy innych sposobach realizacji badań, opisanych w kolejnych punktach (p. 2.3, 2.4).

### 2.2.5. Konfiguracja programu symulacyjnego

Do poprawnej realizacji badań symulacyjnych konieczny jest odpowiedni dobór parametrów symulacji, czyli algorytmu całkowania (ang. solver) jaki realizuje program symulacyjny. Zarówno Matlab/Simulink, jak i Scilab/Xcos dysponują domyślnymi wartościami odpowiednich parametrów (Tab. I-4) i zazwyczaj większość z nich zapewnia poprawną symulację. Wartości parametrów są zapamiętywane w pliku razem ze schematem i można je edytować poprzez menu Simulation/Symulacja w oknie edycji schematu, lub poprzez funkcje podczas uruchamiania symulacji.

Tab. I-4. Domyślne parametry symulacji zapamiętywane razem ze schematem

Matlab/Simulink	Scilab/Xcos
Simulation/Configuration Parameters	Symulacja/Ustawienia
Start time = 0.0 <b>Stop time = 10.0</b>	<b>Final integration time</b> (Ostateczny czas integracji) = <b>1.0E04</b> Realtime scaling (Skalowanie czasu rzeczywistego) = 0.0E00
Solver: ode45 (Type = Variable-step)	Solver 0 (CVODE) - 100 (IDA) = CVode-BFD-Newton
<b>Max step size = auto</b> Min step size = auto Initial step size = auto Relative tolerance = 1e-3 Absolute tolerance = auto	<b>Max step size /0 means no limit/</b> (Maks. rozmiar kroku /0=brak limitu) = 0  Integrator relative tolerance (Integrator względnej tolerancji) = 1.0E-06 Integrator absolute tolerance (Integrator absolutnej tolerancji) = 1.0E-06 Tolerance on time (Tolerancja w czasie) = 1.0E-10 Max iteration time interval (Maks.przedział czasu całkowania) = 1.00001E05

Najczęściej zmieniane parametry to:

- **czas symulacji** – jeśli obiekt jest stabilny, to na wykresach reakcji powinien być widoczny końcowy stan ustalony
  - w Matlabie domyślny czas symulacji (Stop time) jest dość krótki (10) i zwykle trzeba go wydłużyć,
  - w Scilabie domyślny czas symulacji (Final integration time) jest bardzo długi ( $10^4$ ) - na początek lepiej zadać krótszy czas,
- **kroku obliczeń** – małe kroki to przede wszystkim długi czas obliczeń, długie korki to niedokładne obliczenia (wykresy nie są „gładkie”) a czasem niestabilność obliczeniowa (wykres wygląda jak bardzo mocno „zaszumiony”)
  - Matlab domyślnie stosuje zmiennokrokowy algorytm całkowania i maksymalny krok obliczeń (Max step size) dobierany automatycznie (jako 0.001 czasu symulacji) i to często zapewnia poprawną symulację, ale jeśli czas symulacji jest długi, to należy dobrać maksymalny krok (uniezależnić od czasu symulacji),
  - Scilab domyślnie proponuje algorytm stałokrokowy (CVode-BDF-Newton), więc dla przyspieszenia obliczeń można wybrać algorytm zmiennokrokowy np. Runge-Kutta 4(5).

Pozostałe parametry są muszą być dobierane w przypadku badania modeli trudnych obliczeniowo<sup>1</sup>.

W prezentowanych przykładach założono, że czas symulacji jest określany w skrypcie (w Matlabie poprzez funkcje `sim()`, w Scilabie poprzez strukturę `scs_m`), natomiast pozostałe parametry symulacji (parametry algorytmu całkującego) są edytowane (w razie potrzeby) poprzez menu na schemacie.

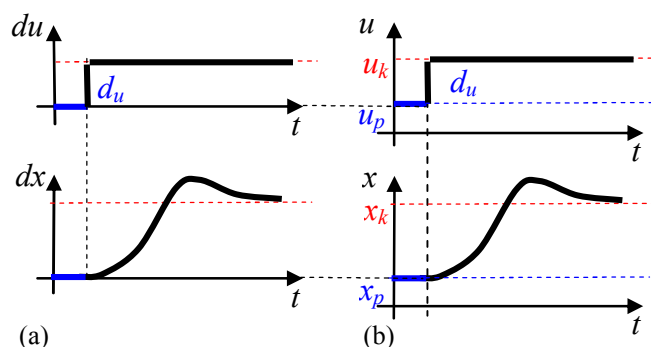
### 2.3. Modele obiektów liniowych „w trybie graficznym” i analiza czasowa

#### 2.3.1. Założenia

Schematy modeli liniowych można tworzyć zarówno ogólną metodą na bazie bloków całkujących (p. 2.2), ale także w prostszy sposób, oparty na możliwości zapisania modelu w postaci macierzowych równań stanu lub transmitancji. Analizowany model obiektu cieplnego (I-2) jest nieliniowy. Jeśli jednak w danych warunkach badawczych można by założyć, że przepływ  $f_p$  jest parametrem obiektu ( $f_p = f_{p0}$ ), to model jest liniowy i można go zapisać w postaci równań stanu lub transmitancji, i tym samym przygotować schemat modelu stosując blok State-Space lub Transfer Fcn.

Schemat będzie sparametryzowany i przygotowany do badania reakcji na skokową zmianę wartości na poszczególnych wejściach ( $q_e$ ,  $T_{zew}$ ) w punkcie pracy ( $T_{wew0}$ ,  $T_{gp0}$ ) wyznaczonym na podstawie początkowych wartości wejściowych ( $Q_{g0}$ ,  $T_{zew0}$ ) oraz parametru  $F_{p0}$  (co umożliwi porównanie rezultatów wprost z wynikami uzyskanymi w p.2.2). Wartości zmiennych wyjściowych ( $T_g$ ,  $T_{wew}$ ) będą zapamiętywane jako wektory wartości ( $aT_{wew}$ ,  $aT_g$ ). Przygotowanie i sterowanie symulacjami ma się odbywać wsadowo.

Ponieważ reakcja modeli liniowych nie zależy od punktu pracy, często wystarczy zrealizowanie prostszych badania – wyznaczenie reakcji na skok w zerowych warunkach początkowych (Rys. I-10a). Uzyskane przebiegi można wówczas interpretować jako reakcję modelu względem punktu równowagi, i w razie potrzeby przesunąć do wybranego punktu równowagi (Rys. I-10b).



Rys. I-10. Badanie reakcji na wymuszenie skokowe (3)

<sup>1</sup> Są to na przykład tak zwane sztywne równania różniczkowe (ang. stiff differential equations) - ich rozwiązania zawierają gładkie i niegładkie rozwiązania, które gwałtownie przechodzą jedno w drugie. Przykład – równanie van der Pola.

### 2.3.2. Równania stanu w środowisku Matlab/Simulink

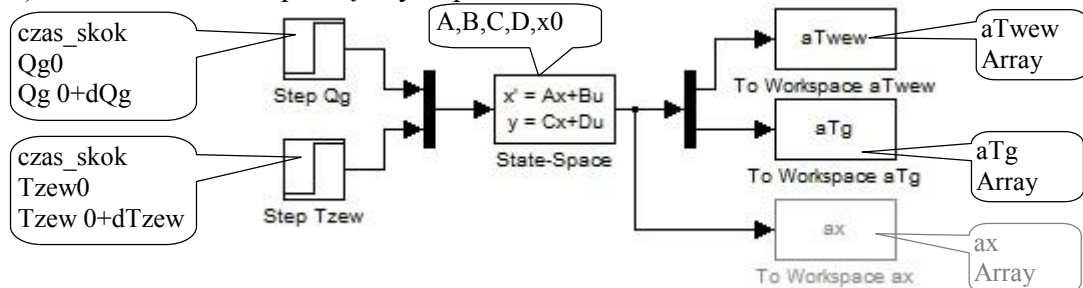
Zlinearyzowaną wersję równań stanu (I-2) przy założeniu stałego przepływu  $f_p$ , można zapisać w postaci macierzowej:

$$\begin{bmatrix} \dot{T}_{wew} \\ \dot{T}_p \end{bmatrix} = \begin{bmatrix} -\frac{(K_g + K_1 + c_{pp}\rho_p f_{p0})}{C_{vw}} & \frac{K_g}{C_{vw}} \\ \frac{K_g}{C_{vg}} & -\frac{K_g}{C_{vg}} \end{bmatrix} \begin{bmatrix} T_{wew} \\ T_p \end{bmatrix} + \begin{bmatrix} 0 & \frac{K_1 + c_{pp}\rho_p f_{p0}}{C_{vw}} \\ \frac{1}{C_{vg}} & 0 \end{bmatrix} \begin{bmatrix} q_g \\ T_{zew} \end{bmatrix} \quad (I-11)$$

Równania (I-11) są podstawą do stworzenia schematu modelu na bazie bloku State-Space (Rys. I-11). Na wejście tego bloku jest podawany wektor zmiennych wejściowych ( $q_g, T_{zew}$ ), a na wyjściu powinien być dostępny wektor zmiennych ( $T_{wew}, T_g$ ), co wymaga zdefiniowania wektora wyjściowego<sup>1</sup>:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} T_{wew} \\ T_p \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q_g \\ T_{zew} \end{bmatrix} \quad (I-12)$$

W dymkach przedstawiono sposób zdefiniowania parametrów w blokach Step (Step time, Initial value, Final value), State-Space (A, B, C, D, Initial conditions) oraz To Workspace<sup>2</sup> (Variable name, Save format). Schemat został zapamiętany w pliku o nazwie *ob2.mdl*.



Rys. I-11. Schemat modelu w oparciu o blok State-Space w środowisku Simulink (*ob2.mdl*)

**Skrypt (*ob2\_skrypt1.m*)** inicjujący zmienne i uruchamiający symulację modelu *ob2.mdl*:

I część (identyfikacja) jest identyczna jak w skrypcie *obl\_skrypt1.m*

```

%===== II część (punkt pracy, parametry, definicje) =====
%warunki początkowe i parametry
Tzew0= TzewN + 0;      %0, 10, 20
Qg0 = QgN * 1.0;      %1.0, 0.7, 0.3
Fp0 = FpN * 1.0;      %1.0, 0.7, 0.3 (parametr)
%-----
%definicja macierzy (u=[Qg; Tzew], x=[Twew; Tp])
A = [-(Kg+K1+cpp*rop*Fp0)/Cvw, Kg/Cvw; ...
     Kg/Cvg, -Kg/Cvg];
B = [0, (K1+cpp*rop*Fp0)/Cvw; ...
     1/Cvg, 0];
C = [1,0; 0,1]; D = [0,0;0,0];
%-----
%stan równowagi
u0=[Qg0; Tzew0];      %wektor zmiennych wejściowych
x0=-A^-1 * B*u0;      %wektor zmiennych stanu
%===== III część (symulacje) =====
czas = 50000;         %czas symulacji
%zakłócenie
czas_skok = 5000;     %moment wystąpienia skoku
dTzew=1;             %0 lub np. 1
dQg=0;               %0 lub np. 0.05*QgN
%-----
%symulacja
[t]=sim('ob2',czas); %t=wektor czasu
%-----
%wykresy
figure, plot(t,aTwew,'r'), grid on, title('Reakcja Twew na skok Tzew');
figure, plot(t,aTg,'r'), grid on, title('Reakcja Tg na skok Tzew');

```

<sup>1</sup> Na wyjściu bloku State-Space dostępny jest tylko wektor wyjściowy  $y$ , definiowany na podstawie zmiennych stanu  $x$  i wymuszeń  $u$ .

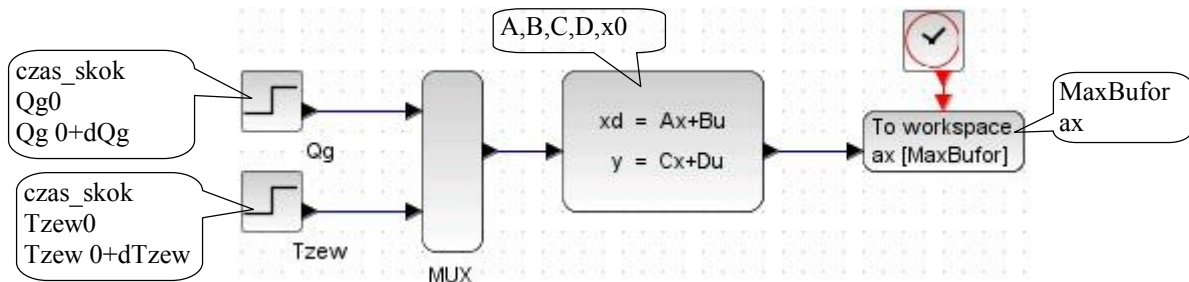
<sup>2</sup> Przedstawiono dwa alternatywne rozwiązania: podział wektora wyjściowego na dwie składowe ( $a_{Twew}$ ,  $a_{Tg}$ ) i zapamiętanie wektora wyjściowego bez dzielenia na składowe ( $ax$ )

```
%wykresy (wersja alternatywna, bez dzielenia wektora wyjściowego na składowe)
%figure, plot(t,ax(:,1),'r'), grid on, title('Reakcja Twew na skok Tzew');
%figure, plot(t,ax(:,2),'r'), grid on, title('Reakcja Tg na skok Tzew');
```

Uruchomienie skryptu spowoduje wygenerowanie reakcji na skok temperatury  $T_{zew}$  o  $1^\circ\text{C}$  przy nominalnych warunkach początkowych (identycznie jak *obl\_skrypt1.m*). Ustalając wartości początkowe zmiennych wejściowych ( $Qg_0$ ,  $Tzew_0$ ) i parametru  $Fg_0$ , można zmienić punkt pracy – ponieważ model jest definiowany za pomocą macierzy, to wykorzystano rozwiązanie w postaci macierzowej (I-6) i możliwość zadawania warunków początkowych w bloku State-Space. Wybór wymuszenia (zmienna i wielkość skoku) następuje przez ustalenie zerowych lub niezerowych wartości skoków ( $dQg$ ,  $dTzew$ ).

### 2.3.3. Równania stanu w środowisku Scilab/Xcos

Podstawą aplikacji równań stanu w środowisku Scilab/Xcos (Rys. I-12) są równania stanu (I-11) i równania wyjściowe (I-12). Zastosowano podstawowe bloki Xcos, a w dymkach przedstawiono parametry zapisane w blokach Step\_Function (Step Time, Initial value, Final value), CLSS (A, B, C, D, Initial state) oraz To Workspace (Size of buffer, Scilab variable name). Schemat został zapamiętany w pliku o nazwie *ob2.xcos*.



Rys. I-12. Schemat modelu w oparciu o blok CLSS w środowisku Xcos (*ob2.xcos*)

**Skrypt (*ob2\_skrypt1.sce*)** inicjujący zmienne i uruchamiający symulację modelu *ob2.xcos*:

I część (identyfikacja) jest identyczna jak w skrypcie *obl\_skrypt1.sce*

```
//===== II czesc (punkt pracy, parametry, definicje) =====
//warunki początkowe i parametry
Tzew0= TzewN + 0; //0, 10, 20
Qg0 = QgN * 1.0; //1.0, 0.7, 0.3
Fp0 = FpN * 1.0; //1.0, 0.7, 0.3 (parametr)
//-----
//definicja macierzy (u=[Qg; Tzew], x=[Tzew; Tp])
A = [-(Kg+Kl+cpp*rop*Fp0)/Cvw, Kg/Cvw, ...
      Kg/Cvg, -Kg/Cvg];
B = [0, (Kl+cpp*rop*Fp0)/Cvw, ...
      1/Cvg, 0];
C = [1,0; 0,1]; D = [0,0;0,0];
//-----
//stan równowagi
u0=[Qg0; Tzew0]; //wektor zmiennych wejściowych
x0=-A^-1 * B*u0; //%wektor zmiennych stanu
//===== III czesc (symulacje) =====
loadXcosLibs(); //wczytanie bibliotek xcos
importXcosDiagram(C:\projekty\ob3.xcos); //wczytanie modelu (tworzy zmienną scs_m)
czas = 50000; zegar=2; //czas symulacji i parametr: Clock/Period
MaxBufor = czas*10; //parametr To workspace/Size of buffer (było czas/zegar)
scs_m.props.tf=czas; //czas trwania symulacji (zamiast czasu ze schematu)
//zakłócenie
czas_skok = 5000; //moment wystąpienia skoku
dTzew=1; //0 lub 1
dQg=0; //0 lub 0.05*QgN
//-----
//symulacja
scicos_simulate(scs_m); //czas symulacji: scs_m.props.tf
//wykresy
figure , plot(ax.time,ax.values(:,1),red'), xgrid(2), title('Reakcja Twew na skok Tzew');
figure , plot(ax.time,ax.values(:,2),red'), xgrid(2), title('Reakcja Tg na skok Tzew');
```

Uwagi do skryptu analogiczne jak do *obl\_skrypt1.sce*

### 2.3.4. Transmitancje w środowisku Matlab/Simulink

Zlinearyzowana wersja modelu (I-2) ze stałym przepływem  $f_p$ , umożliwia obliczenie następujących transmitancji:

$$T_{zew}(s) = G_{11}(s)q_g(s) + G_{12}(s)T_{zew}(s) = \frac{K_g}{M_1M_2 - K_g^2}q_g(s) + \frac{(K_1 + c_{pp}\rho_p f_{p0})M_1}{M_1M_2 - K_g^2}T_{zew}(s)$$

$$T_g(s) = G_{21}(s)q_g(s) + G_{22}(s)T_{zew}(s) = \frac{M_2}{M_1M_2 - K_g^2}q_g(s) + \frac{K_g(K_1 + c_{pp}\rho_p f_{p0})}{M_1M_2 - K_g^2}T_{zew}(s) \quad (I-13)$$

gdzie  $M_1 = C_{vg}s + K_g$ ,  $M_2 = C_{vw}s + K_g + K_1 + c_{pp}\rho_p f_{p0}$ ,

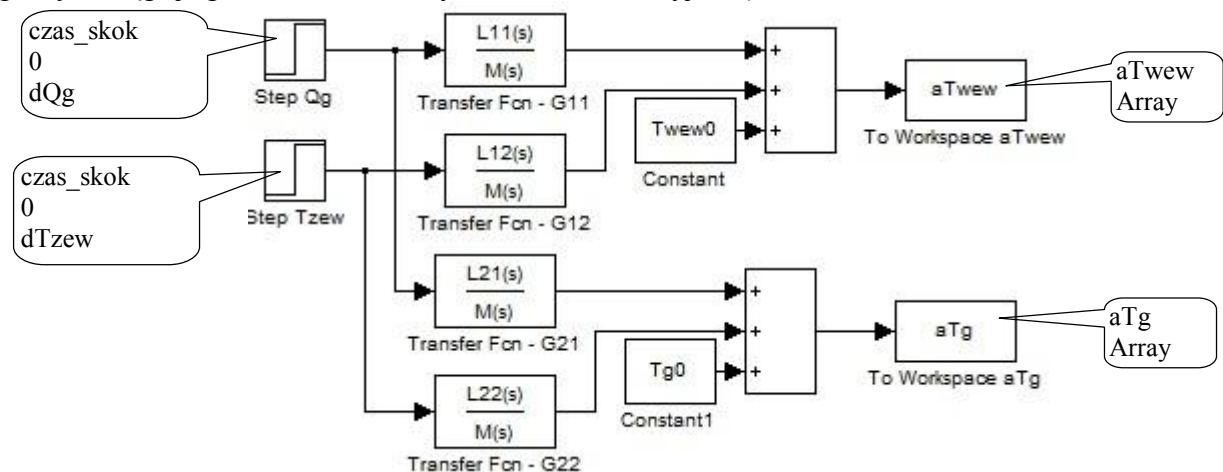
lub po rozwinięciu i uporządkowaniu wyrażeń:

$$T_{zew}(s) = \frac{K_g}{M}q_g(s) + \frac{(K_1 + c_{pp}\rho_p f_{p0})(C_{vg}s + K_g)}{M}T_{zew}(s)$$

$$T_g(s) = \frac{C_{vw}s + K_g + K_1 + c_{pp}\rho_p f_{p0}}{M}q_g(s) + \frac{K_g(K_1 + c_{pp}\rho_p f_{p0})}{M}T_{zew}(s) \quad (I-14)$$

gdzie  $M = C_{vg}C_{vw}s^2 + [C_{vg}(K_g + K_1 + c_{pp}\rho_p f_{p0}) + C_{vw}K_g]s + K_g(K_1 + c_{pp}\rho_p f_{p0})$ .

Na podstawie transmitancji (I-14) zrealizowano schemat (Rys. I-13), zapamiętany następnie w pliku o nazwie *ob3.mdl*. Parametrami bloków Transfer Fcn (Numerator coefficients, Denominator coefficients) są odpowiednie wektory współczynników transmitancji (L11, L12, L21, L22, M) zdefiniowane w skrypcie. Parametry pozostałych bloków Step (Step time, Initial value, Final value) i To Workspace (Variable name, Save format) przedstawiono w dymkach. Parametry wprowadzone na schemacie uwzględniają fakt, że transmitancja jest z definicji opisem dynamiki przy założeniu zerowych warunków początkowych (stąd wartości początkowe w blokach Step wynoszą 0, a ustalenie punktu pracy następuje przez dodanie stałych wartości na wyjściu).



Rys. I-13. Schemat modelu w oparciu o bloki Transfer Fcn w środowisku Simulink (*ob3.mdl*)

**Skrypt** (*ob3\_skrypt1.m*) inicjujący zmienne i uruchamiający symulację modelu *ob3.mdl*:

I część (identyfikacja) jest identyczna jak w skrypcie *ob1\_skrypt1.m*

```

§===== II część (punkt pracy, parametry, definicje) =====
§warunki początkowe i parametry
Tzew0= TzewN + 0;    §0, 10, 20
Qg0 = QgN * 1.0;    §1.0, 0.7, 0.3
Fp0 = FpN * 1.0;    §1.0, 0.7, 0.3 (parametr)
§-----
§definicja współczynników transmitancji (G11=Twew/Qg,G12=Twew/Tzew,G21=Tg/Qg,G22=Tg/Tzew)
M = [Cvg*Cvw, Cvg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg, Kg*(K1+cpp*rop*Fp0)];
L11 = [Kg];
L12 = [Cvg*(K1+cpp*rop*Fp0), Kg*(K1+cpp*rop*Fp0)];
L21 = [Cvw, Kg+K1+cpp*rop*Fp0];
L22 = [Kg*((K1+cpp*rop*Fp0))];
§-----
§stan równowagi
Twew0 = Qg0/(K1+cpp*rop*Fp0)+Tzew0;
Tg0 = Qg0/Kg + Twew0;

```

```

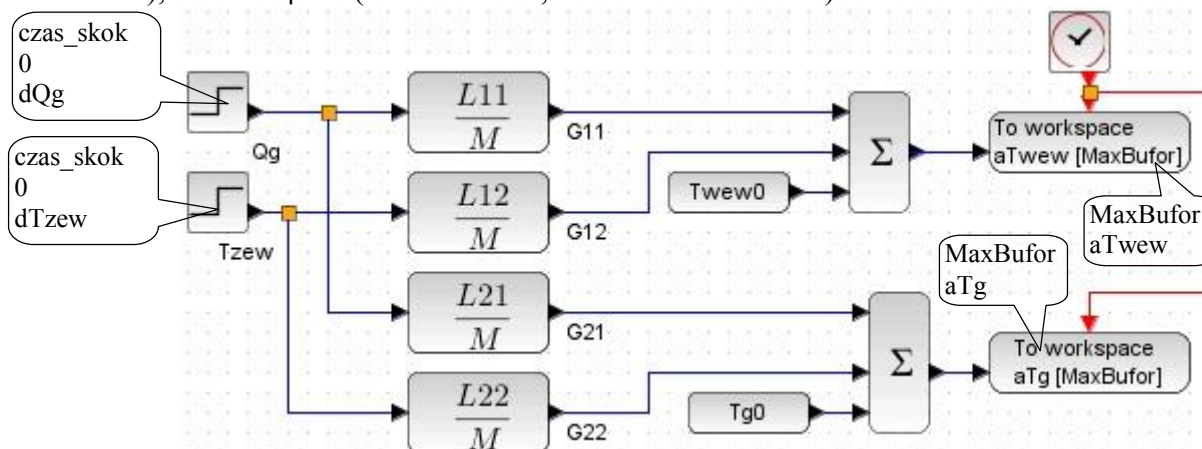
===== III część (symulacje) =====
czas = 50000;          %czas symulacji
%zakłócenie
czas_skok = 5000;     %moment wystąpienia skoku
dTzew=1;             %0 lub np. 1
dQg=0;               %0 lub np. 0.1*QgN
%-----
%symulacja
[t]=sim('ob3',czas); %t=wektor czasu
%-----
%wykresy
figure, plot(t,aTwew,'r'), grid on, title('Reakcja Twew na skok Tzew');
figure, plot(t,aTg,'r'), grid on, title('Reakcja Tg na skok Tzew');

```

Uruchomienie skryptu spowoduje wygenerowanie reakcji na skok temperatury  $T_{zew}$  o  $1^\circ\text{C}$  przy nominalnych warunkach początkowych (identycznie jak *obl\_skrypt1.m*). Punkt pracy, obliczony na podstawie wartości początkowych zmiennych wejściowych ( $Qg_0$ ,  $Tzew_0$ ) i parametru  $Fg_0$ , jest uwzględniany przez przesunięcie sygnałów wyjściowych z bloków Transfer Fcn. Wybór wymuszenia (zmienna i wielkość skoku) następuje przez ustalenie zerowych lub niezerowych wartości skoków ( $dQg$ ,  $dTzew$ ).

### 2.3.5. Transmitancje w środowisku Scilab/Xcos

Schemat (Rys. I-14) został zrealizowany na podstawie transmitancji (I-14) i zapamiętany w pliku o nazwie *ob3.xcos*. Jako parametry bloków transmitancji CLR (Numerator coefficients, Denominator coefficients) należy użyć wielomianów ( $L11$ ,  $L12$ ,  $L21$ ,  $L22$ ,  $M$ ) – zdefiniowano je w skrypcie. Parametry pozostałych bloków przedstawiono w dymkach: Step Function (Step Time, Initial value, Final value), To Workspace (Size of buffer, Scilab variable name).



Rys. I-14. Schemat modelu w oparciu o bloki CLR w środowisku Xcos (*ob3.xcos*)

**Skrypt** (*ob3\_skrypt1.sce*) inicjujący zmienne i uruchamiający symulację modelu *ob3.xcos*:

I część (identyfikacja) jest identyczna jak w skrypcie *obl\_skrypt1.sce*

```

===== II czesc (punkt pracy, parametry, definicje) =====
//warunki początkowe i parametry
Tzew0= TzewN + 0; //0, 10, 20
Qg0 = QgN * 1.0; //1.0, 0.7, 0.3
Fp0 = FpN * 1.0; //1.0, 0.7, 0.3 (parametr)
%-----
//definicja wielomianow transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
s=poly(0,'s'); //definicja zmiennej s
M = Cvlg*Cvw*s^2 + (Cvlg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg)*s + Kg*(K1+cpp*rop*Fp0);
L11 = Kg;
L12 = Cvlg*(K1+cpp*rop*Fp0)*s + Kg*(K1+cpp*rop*Fp0);
L21 = Cvw*s + Kg+K1+cpp*rop*Fp0;
L22 = Kg*(K1+cpp*rop*Fp0);
%-----
//stan równowagi
Twew0 = Qg0/(K1+cpp*rop*Fp0)+Tzew0;
Tg0 = Qg0/Kg + Twew0;

```



```

===== III czesc (symulacje) =====
loadXcosLibs();           //wczytanie bibliotek xcos
importXcosDiagram('C:\projekty\ob3\xcos'); //wczytanie modelu (tworzy zmienna scs_m)
czas = 50000; zegar=2;    //czas symulacji i parametr: Clock/Period
MaxBufor = czas*10;      //parametr To workspace/Size of buffer (bylo czas/zegar)
scs_m.props.tf=czas;     //czas trwania symulacji (zamiast czasu ze schematu)
//zakłócenie
czas_skok = 5000;        //moment wystapienia skoku
dTzew=1;                 //0 lub 1
dQg=0;                   //0 lub 0.05*QgN
//-----
//symulacja
scicos_simulate(scs_m);  //czas symulacji: scs_m.props.tf
//-----
//wykresy
figure, plot(aTwew.time,aTwew.values,'red'), xgrid(2), title('Reakcja Twew na skok Tzew');
figure, plot(aTg.time,aTg.values,'red'), xgrid(2), title('Reakcja Tg na skok Tzew');

```

Uwagi do skryptu analogiczne jak do *obl\_skrypt1.sce*

## 2.4. Modele obiektów liniowych „w trybie tekstowym”

### 2.4.1. Założenia

Kolejny sposób realizowania podstawowych badań dynamiki odbywa się bez rysowania schematów (bez wykorzystywania przybornika Matlab/Simulink, lub funkcji Scilab/Xcos). Zarówno model, jak i program badań będą realizowane w skrypcie. Jeśli model jest liniowy (można go zapisać w postaci równań stanu lub transmitancji), to wówczas najprostszym sposobem jest zastosowanie specjalistycznych funkcji, dedykowanych do badania dynamiki liniowych obiektów SISO/MIMO, w tym do definiowania modeli i generowania różnego typu wykresów (między innymi odpowiedzi skokowej). W tych warunkach bardzo prosto realizuje się pełne badanie modeli MIMO. Badany model (I-11)/(I-14) jest typu MIMO (2 wejścia i 2 wyjścia) - wszystkie przedstawione skrypty realizują pełne badanie odpowiedzi skokowych modelu w zerowych warunkach początkowych, czyli wygenerowanie reakcji każdego wyjścia na skok jednostkowy na każdym z wejść.

W przypadku Matlaba funkcje te znajdują się w przyborniku Control. Modele (równania stanu i transmitancje) zdefiniowane za pomocą odpowiednich funkcji (*ss*, *tf*) są przechowywane w przestrzeni roboczej Matlaba jako obiekty LTI (Linear Time-Invariant System objects). Funkcja do generowania odpowiedzi skokowej (*step*) obsługuje modele SISO i MIMO. Natomiast w przypadku Scilaba zarówno funkcja do definicji modeli liniowych (*syslin*), jak i funkcja do generowania wykresów (*csim*) są zawarte w pakiecie (w ramach realizacji funkcjonalności CACSD). Można definiować modele MIMO, ale generowanie wykresów ogranicza się do obiektów SISO i SIMO.

### 2.4.2. Równania stanu w środowisku Matlab/Control

Podstawą definicji modelu opisanego równaniami (I-11) i (I-12) jest w Matlabie zastosowanie funkcji *ss()*. Przedstawiony skrypt jest najprostszym sposobem wygenerowania odpowiedzi na wymuszenia skokowe za pomocą funkcji *step()*.

**Skrypt** (*ob2\_skrypt2.m*) inicjujący zmienne, definiujący model i uruchamiający symulację:

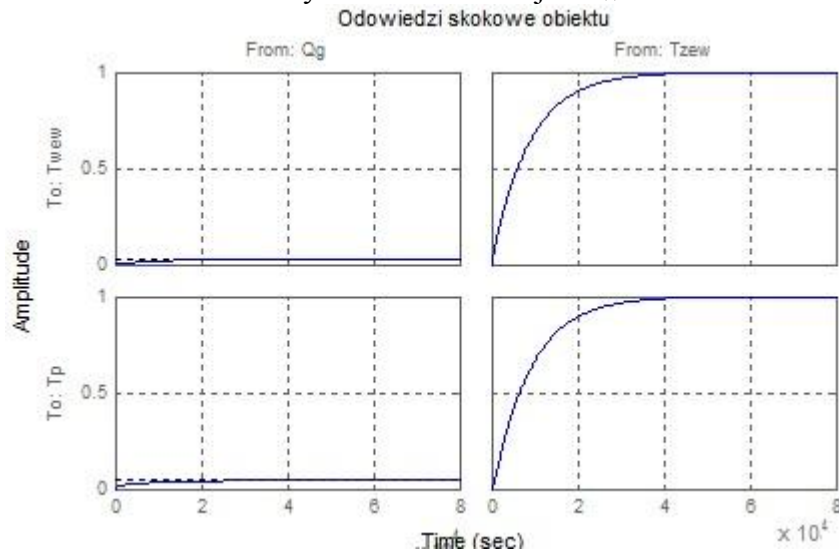
I część (identyfikacja) jest identyczna jak w skrypcie *obl\_skrypt1.m*

```

%===== II część (parametry, definicje) =====
%parametry
Fp0 = FpN * 1.0;      % np.: 1.0, 0.7, 0.3 (parametr)
%-----
%definicja macierzy (u=[Qg; Tzew], x=[Twew; Tp]) i modelu ob2
A = [-(Kg+K1+cpp*rop*Fp0)/Cvw, Kg/Cvw; ...
     Kg/Cvg, -Kg/Cvg];
B = [0, (K1+cpp*rop*Fp0)/Cvw; ...
     1/Cvg, 0];
C = [1, 0; 0, 1]; D = [0, 0; 0, 0];
%-----
%definicja modelu MIMO
%ob2 = ss(A, B, C, D); %definicja podstawowa (bez nazw)
ob2 = ss(A, B, C, D, 'InputName', ['Qg ' ; 'Tzew'], 'OutputName', ['Twew'; 'Tp ']);
%===== III część (symulacje) =====
%symulacja i wykresy
step(ob2)
title('Odwiedzi skokowe obiektu');

```

Wykonanie funkcji `step()` realizuje pełne badanie odpowiedzi skokowych modelu w zerowych warunkach początkowych (Rys. I-15). Wykresy są opisane za pomocą nazw zmiennych dzięki wykorzystaniu możliwości definiowania tych nazw w funkcji `ss()`.



Rys. I-15. Wynik działania skryptu „ob2\_skrypt2.m” (wykresy po sformatowaniu – patrz Załącznik B.3)

Uniwersalne zakłócenie w postaci skoku jednostkowego zazwyczaj nie jest najlepszym rozwiązaniem w przypadku modeli obiektów fizycznych. W analizowanym przypadku zmiana mocy  $Q_g$  o 1W to zaledwie 0.07% zakresu zmienności mocy ( $0 \div Q_{gN}$ ), natomiast zmiana temperatury  $T_{zew}$  o  $1^\circ\text{C}$ , to 2.5% zakresu zmienności temperatury ( $T_{zewN} \approx 20^\circ\text{C}$ ). Ponieważ badany model jest liniowy to wykresy można skalować i przesuwac (patrz Załącznik B.4), tak aby uzyskać reakcje na dowolne skoki zmiennych wejściowych, w dowolnym punkcie pracy, to znaczy badania analogiczne jak w punkcie 2.3.

### 2.4.3. Równania stanu w środowisku Scilab

Wykorzystanie równań (I-11) i (I-12) do definicji modelu w Scilabie polega na zastosowaniu funkcji `syslin()`, natomiast wygenerowanie odpowiedzi skokowej modelu odbywa się za pomocą funkcji `csim()`. Wynikiem działania funkcji `syslin()` jest model typu MIMO, ale parametrem funkcji `csim()` może być jedynie model SISO lub SIMO, dlatego zastosowano konwersję modelu na transmitancje i generowanie odpowiedzi skokowej dla każdej z nich.

**Skrypt** (`ob2_skrypt2.sce`) inicjujący zmienne, definiujący model i uruchamiający symulację:

I część (identyfikacja) jest identyczna jak w skrypcie `ob1_skrypt1.sce`

```

//===== II część (punkt pracy, parametry, definicje) =====
//parametry
Fp0 = FpN * 1.0; //np.:1.0, 0.7, 0.3 (parametr)
//-----
//definicja macierzy (u=[Qg; Tzew], x=[Twew; Tp]) i modelu ob2
A = [-(Kg+K1+cpp*rop*Fp0)/Cvw, Kg/Cvw; ...
      Kg/Cvg, -Kg/Cvg];
B = [0, (K1+cpp*rop*Fp0)/Cvw; ...
      1/Cvg, 0];
C = [1,0; 0,1]; D = [0,0; 0,0];
//-----
//definicja modelu MIMO
ob2 = syslin('c',A,B,C,D);
ob2t = ss2tf(ob2); //konwersja na 4 transmitancje
//===== III część (symulacje) =====
//symulacja i wykresy
t = 0:1:50000;
subplot(2,2,1), plot(t, csim('step',t,ob2t(1,1))); xgrid(2), title('Twew od Qg');
subplot(2,2,2), plot(t, csim('step',t,ob2t(1,2))); xgrid(2), title('Twew od Tzew');
subplot(2,2,3), plot(t, csim('step',t,ob2t(2,1))); xgrid(2), title('Tp od Qg');
subplot(2,2,4), plot(t, csim('step',t,ob2t(2,2))); xgrid(2), title('Tp od Tzew');

```

Wartości wygenerowane za pomocą funkcji `csim()` można zapamiętać, przeskalować i przesunąć (patrz Załącznik B.4), tak aby uzyskać reakcje na dowolne skoki zmiennych wejściowych, w dowolnym punkcie pracy.

#### 2.4.4. Transmitancje w środowisku Matlab/Control

Definicję modelu w postaci transmitancji w Matlabie można zrealizować w różny sposób, na przykład: za pomocą funkcji `tf()` lub `zpk()`, jako obiekt MIMO lub zestaw obiektów SISO. Poniżej przedstawiono dwa warianty skryptu, które wykorzystują funkcję `tf()` i realizują takie same badania jak skrypt `ob2_skrypt2.m`, tzn. wszystkie odpowiedzi skokowe modelu.

**Skrypt** (`ob3_skrypt2a.m`) inicjujący zmienne, definiujący model MIMO i uruchamiający symulację:

Pierwszy wariant skryptu bazuje na wzorach (I-14), analogicznie jak schemat wykorzystujący bloki Transfer Fcn (Rys. I-13). Wszystkie cztery transmitancje są zdefiniowane jako jeden obiekt LTI (analogicznie jak w skrypcie `ob2_skrypt2.m`).

I część (identyfikacja) jest identyczna jak w skrypcie `ob1_skrypt1.m`

```

%===== II czesc (parametry, definicje) =====
%parametry
Fp0 = FpN * 1.0;    % np.: 1.0, 0.7, 0.3 (parametr)
%-----
%definicja współczynników transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
M = [Cvg*Cvw, Cvg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg, Kg*(K1+cpp*rop*Fp0)];
L11 = [Kg];
L12 = [Cvg*(K1+cpp*rop*Fp0), Kg*(K1+cpp*rop*Fp0)];
L21 = [Cvw, Kg+K1+cpp*rop*Fp0];
L22 = [Kg*(K1+cpp*rop*Fp0)];
%-----
%definicja modelu MIMO
ob3 = tf({L11,L12; L21,L22}, {M,M; M,M});

%===== III czesc (symulacje) =====
%symulacja i wykresy
step(ob3)
title('Odpowiedzi skokowe obiektu');

```

**Skrypt** (`ob3_skrypt2b.m`) inicjujący zmienne, definiujący modele SISO i uruchamiający symulację:

Drugi wariant skryptu również bazuje na wzorach (I-14), ale funkcja `tf()` jest użyta czterokrotnie – każda transmitancja badanego modelu jest oddzielnym obiektem LTI.

I część (identyfikacja) jest identyczna jak w skrypcie `ob1_skrypt1.m`

```

%===== II część (parametry, definicje) =====
%parametry
Fp0 = FpN * 1.0;    % np.: 1.0, 0.7, 0.3 (parametr)
%-----
%definicja transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
M = [Cvg*Cvw, Cvg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg, Kg*(K1+cpp*rop*Fp0)];
ob3_G11 = tf([Kg], M);
ob3_G12 = tf([Cvg*(K1+cpp*rop*Fp0), Kg*(K1+cpp*rop*Fp0)], M);
ob3_G21 = tf([Cvw, Kg+K1+cpp*rop*Fp0], M);
ob3_G22 = tf([Kg*(K1+cpp*rop*Fp0)], M);

%===== III część (symulacje) =====
%symulacja i wykresy
figure
subplot(221), step(ob3_G11); title('Twew od Qg');
subplot(222), step(ob3_G12); title('Twew od Tzew');
subplot(223), step(ob3_G21); title('Tg od Qg');
subplot(224), step(ob3_G22); title('Tg od Tzew');

```

Dostępny jest też sposób definiowania transmitancji za pomocą wyrażeń ze zmienną  $s$ , utworzoną wcześniej za pomocą funkcji `tf()`. Można wówczas definiować modele na podstawie wzorów (I-14):

```

%definicja transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
s=tf('s');
M = Cvg*Cvw*s^2 + ( Cvg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg ) * s + Kg*(K1+cpp*rop*Fp0);
ob3_G11 = Kg / M;
ob3_G12 = ( Cvg*(K1+cpp*rop*Fp0)*s + Kg*(K1+cpp*rop*Fp0) ) / M;
ob3_G21 = ( Cvw*s + Kg+K1+cpp*rop*Fp0 ) / M;
ob3_G22 = Kg*(K1+cpp*rop*Fp0) / M;

```

lub wzorów (I-13):

```

$definicja transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
s=tf('s');
M1 = Cvg*s + Kg;
M2 = Cvw*s + Kg+K1+cpp*rop*Fp0;
M = M1*M2-Kg^2
ob3_G11 = Kg / M;
ob3_G12 = (K1+cpp*rop*Fp0) * M1 / M;
ob3_G21 = M2 / M;
ob3_G22 = Kg*(K1+cpp*rop*Fp0) / M;

```

Wykresy odpowiedzi skokowych dla transmitancji zdefiniowanych w skryptach można wygenerować za pomocą funkcji `step()`. Można je również skalować i przesuwając (patrz Załącznik B.4).

### 2.4.5. Transmitancje w środowisku Scilab

W Scilabie do definiowania modelu w postaci transmitancji wykorzystywana jest funkcja `simlin()`, z parametrami w postaci wielomianów zmiennej  $s$ . Skrypt bazuje na wzorach (I-14) i definiuje model jako zestaw czterech obiektów SISO, które posłużą do wygenerowania odpowiedzi skokowej modelu za pomocą funkcji `csim()`.

**Skrypt** (*ob3\_skrypt2b.sce*) inicjujący zmienne, definiujący modele SISO i uruchamiający symulację:

I część (identyfikacja) jest identyczna jak w skrypcie *ob1\_skrypt1.m*

```

//===== II czesc (punkt pracy, parametry, definicje) =====
//parametry
Fp0 = FpN * 1.0; //np.:1.0, 0.7, 0.3 (parametr)
//-----
//definicja transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
s=poly(0,'s'); //definicja zmiennej s
M = Cvg*Cvw*s^2 + ( Cvg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg ) * s + Kg*(K1+cpp*rop*Fp0);
ob3_G11 = syslin('c',Kg, M);
ob3_G12 = syslin('c', Cvg*(K1+cpp*rop*Fp0)*s + Kg*(K1+cpp*rop*Fp0), M);
ob3_G21 = syslin('c', Cvw*s + Kg+K1+cpp*rop*Fp0, M);
ob3_G22 = syslin('c', Kg*(K1+cpp*rop*Fp0), M);

//===== III czesc (symulacje) =====
//symulacja i wykresy
t = 0:1:50000;
subplot(2,2,1), plot(t, csim('step',t,ob3_G11)); xgrid(2), title('Twew od Qg');
subplot(2,2,2), plot(t, csim('step',t,ob3_G12)); xgrid(2), title('Twew od Tzew'); ;
subplot(2,2,3), plot(t, csim('step',t,ob3_G21)); xgrid(2), title('Tp od Qg');
subplot(2,2,4), plot(t, csim('step',t,ob3_G22)); xgrid(2), title('Tp od Tzew');

```

Ponieważ do definiowania transmitancji zawsze potrzebne są wielomiany zmiennej  $s$ , to można zdefiniować model wprost na podstawie wzorów (I-14):

```

//definicja transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
s=poly(0,'s'); //definicja zmiennej s
M = Cvg*Cvw*s^2 + ( Cvg*(Kg+K1+cpp*rop*Fp0)+Cvw*Kg ) * s + Kg*(K1+cpp*rop*Fp0);
ob3_G11 = Kg / M;
ob3_G12 = ( Cvg*(K1+cpp*rop*Fp0)*s + Kg*(K1+cpp*rop*Fp0) ) / M;
ob3_G21 = ( Cvw*s + Kg+K1+cpp*rop*Fp0 ) / M;
ob3_G22 = Kg*(K1+cpp*rop*Fp0) / M;

```

lub wzorów (I-13):

```

//definicja transmitancji (G11=Twew/Qg, G12=Twew/Tzew, G21=Tg/Qg, G22=Tg/Tzew)
s=poly(0,'s'); //definicja zmiennej s
M1 = Cvg*s + Kg;
M2 = Cvw*s + Kg+K1+cpp*rop*Fp0;
M = M1*M2-Kg^2
ob3_G11 = Kg / M;
ob3_G12 = (K1+cpp*rop*Fp0) * M1 / M;
ob3_G21 = M2 / M;
ob3_G22 = Kg*(K1+cpp*rop*Fp0) / M;

```

Przykład skalowania i przesuwania wykresów wygenerowanych za pomocą funkcji `csim()` przedstawiono w Załącznikach (B.4).

## 2.5. Modele obiektów liniowych/nieliniowych w pliku funkcyjnym

### 2.5.1. Założenia

Metoda wykorzystuje tylko elementarne funkcje programu symulacyjnego (np. Matlab bez dodatkowych przyborników).

### 2.5.2. Równania różniczkowe w pliku funkcyjnym Matlab

### 2.5.3. Równania różniczkowe w pliku funkcyjnym Scilab

## 3. Inne metody i badania

### 3.1. Inne charakterystyki

#### 3.1.1. Charakterystyki statyczne

Typowym sposobem generowania charakterystyk statycznych jest wykorzystanie rozwiązania (I-4) układu równań statycznych. Wzór ten był wykorzystywany we wcześniejszych skryptach do obliczania punktu równowagi (*obl\_skrypt1* i *ob3\_skrypt1*). Kolejne skrypty ilustrują zastosowanie wzorów do generowania charakterystyk statycznych, na przykładzie wykresów  $T_{wew}(Q_g)$  i  $T_g(Q_g)$  przy założeniu, że pozostałe zmienne wejściowe mają wartości nominalne. Wzory (I-4) zostaną zapisane z użyciem operatorów z kropką, co pozwala wykonać operacje na elementach wektorów – wektorem będą wartości zmiennej  $Q_g$  w zakresie od 0 do wartości nominalnej  $Q_{gN}$ .

**Skrypt** (*obl\_stat.m*) charakterystyki statyczne  $T_{wew}(Q_g)$  i  $T_g(Q_g)$ ; Matlab

I część (identyfikacja) jest identyczna jak w skrypcie *obl\_skrypt1.m*

```
Tzew0= TzewN;
Qg0 = [0:0.1*QgN:QgN];
Fp0 = FpN;
Twew0 = Qg0./(K1+cpp*rop.*Fp0)+Tzew0;           %wektor wartości Twew
Tg0 = Qg0./Kg + Twew0;                          %wektor wartości Tg
subplot(211), hold on, grid on,
plot(Qg0, Twew0);plot(QgN, TwewN, 'ro'); title('Twew(Qg), TzewN, FgN');
subplot(212), hold on, grid on,
plot(Qg0, Tg0);plot(QgN, TgN, 'ro'); title('Tg(Qg), TzewN, FgN');
```

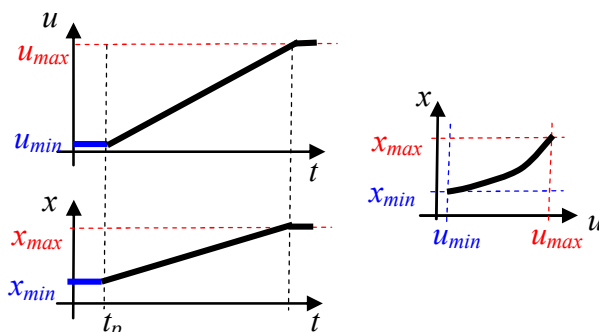
**Skrypt** (*obl\_stat.sce*) charakterystyki statyczne  $T_{wew}(Q_g)$  i  $T_g(Q_g)$ , Scilab

I część (identyfikacja) jest identyczna jak w skrypcie *obl\_skrypt1.sce*

```
Tzew0= TzewN;
Qg0 = [0:0.1*QgN QgN];
Fp0 = FpN;
Twew0 = Qg0./(K1+cpp*rop.*Fp0)+Tzew0;           //wektor wartości Twew
Tg0 = Qg0./Kg + Twew0;                          //wektor wartości Tg
subplot(231), xgnd(2);
plot(Qg0, Twew0); plot(QgN, TwewN, 'ro'); title('Twew(Qg), TzewN, FgN');
subplot(234), xgnd(2);
plot(Qg0, Tg0); plot(QgN, TgN, 'ro'); title('Tg(Qg), TzewN, FgN');
```

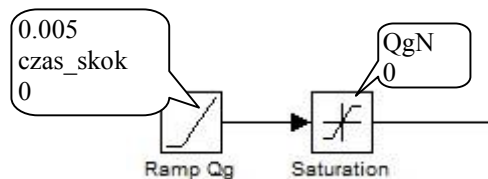
Dodatkowo na wykresach wpisano punkt o wartościach nominalnych (czerwone kółko). Jeśli punkt ten znajduje się na linii charakterystyki, to potwierdza poprawność wygenerowanych wykresów. Analogicznie powstają charakterystyki w funkcji pozostałych wejść. Ponieważ model ma kilka wejść, to zwykle generuje się rodziny charakterystyk (patrz Załącznik B.2). Znaczenie użytkowe mają przede wszystkim charakterystyki w zakresie pracy (działania) danego obiektu, czyli wartości, które są osiągalne w rzeczywistych warunkach.

Charakterystyki statyczne można również otrzymać wykorzystując schemat modelu, wykonując eksperymenty analogiczne jak podczas zdejmowania charakterystyk na rzeczywistych stabilnych obiektach – zmieniamy wartość na wybranym wejściu i odczytujemy wartości na wyjściach po osiągnięciu stanu równowagi, i tak dla kilku wartości wejściowych. Rys. I-16 przedstawia zautomatyzowaną wersję tego eksperymentu.



Rys. I-16. Eksperymentalna charakterystyka statyczna

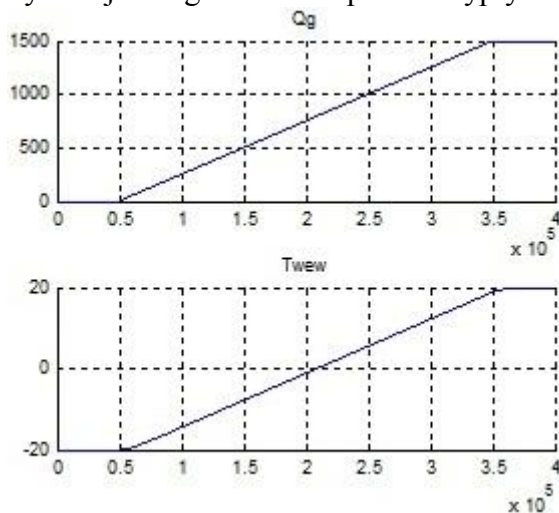
Rozpoczyna się on od stanu równowagi ( $u_{min}, x_{min}$ ), a następnie na wejście  $u$  podawany jest sygnał bardzo wolno narastający, na tyle wolno, że układ nadąża się stabilizować. Warunki eksperymentu są spełnione, jeśli w momencie, gdy sygnał  $u$  przestanie narastać, to sygnał wyjściowy  $x$  również przestanie się zmieniać. Charakterystyka statyczna  $x(u)$  powstaje na podstawie zarejestrowanych wartości  $x$  i  $u$ . Do generowania narastającego sygnału służy blok Ramp (Matlab/Scilab).



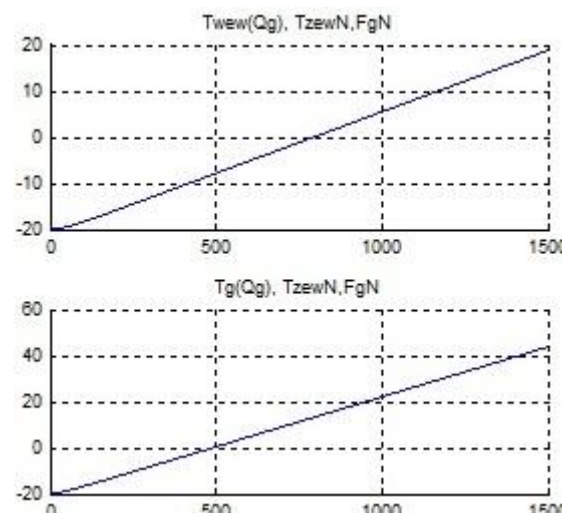
Rys. I-17. Sygnał  $Q_g$  narastający ( $0 \div Q_g N$ )

Na Rys. I-17 przedstawiono sposób realizacji sygnału  $Q_g$  który zastąpi blok Step  $Q_g$  na schemacie Rys. I-7, podczas symulacji przygotowującej dane dla charakterystyk  $T_{wew}(Q_g)$  i  $T_g(Q_g)$ . W dymkach widoczne są parametry bloku Ramp (Slope, Start time, Initial output) i Saturation (Upper limit, Lower limit). Nachylenie (Slope) zostało dobierane eksperymentalnie. Przebiegi uzyskane podczas symulacji

(Rys. I-18) spełniają warunki poprawnego eksperymentu, więc uzyskane charakterystyki (Rys. I-19) są identyczne jak te generowane przez skrypty *obl\_stat*.



Rys. I-18. Przebiegi z symulacji dla narastającego  $Q_g$



Rys. I-19. Charakterystyki statyczne  $T_{wew}(Q_g)$  i  $T_g(Q_g)$

### 3.1.2. Odpowiedzi impulsowe

W rzeczywistych warunkach badanie reakcji obiektu na impulsową zmianę wartości wejścia jest stosowane wówczas, gdy nie można wykonać wymuszenia skokowego (np. ze względu na ograniczenia technologiczne). W przypadku obiektów liniowych istnieje prosta zależność pomiędzy odpowiedzią impulsową  $k(t)$  i odpowiedzią skokową  $h(t)$  tego obiektu - ponieważ wymuszenie impulsowe  $\delta(t)$  jest pochodną wymuszenia skokowego  $1(t)$ , to odpowiedź impulsowa  $k(t)$  jest pochodną odpowiedzi skokowej  $h(t)$ :

$$k(t) = \dot{h}(t) \tag{I-15}$$

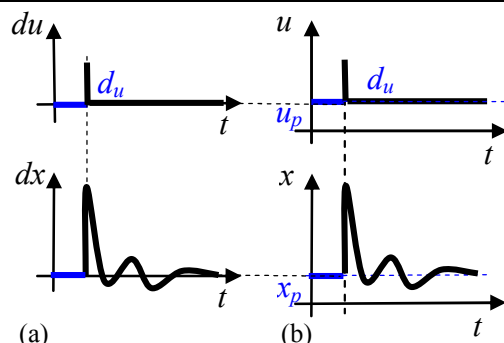
Zależność ta jest wykorzystywana w badaniach – można wykonać tylko jedno z badań (odpowiedź skokową lub impulsową), a drugą obliczyć.

Najłatwiej jest wygenerować symulacyjnie reakcje liniowych modeli na wymuszenie impulsowe za pomocą specjalistycznej funkcji, analogicznie jak w przypadku odpowiedzi skokowej (p. 2.4.2÷2.4.5):

	Matlab/Control	Scilab
Odpowiedź skokowa	<code>step(obiekt);</code>	<code>csim('step', t, obiekt)</code>
Odpowiedź impulsowa	<code>impulse(obiekt);</code>	<code>csim('impuls', t, obiekt)</code>

Funkcje te realizują proste badania w zerowych warunkach początkowych (Rys. I-20a), ale wyniki można przesunąć do wybranego punktu równowagi (Rys. I-20b).

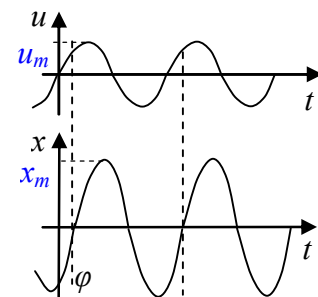
Odpowiedź impulsowa obiektu, to reakcja na idealny impuls  $\delta(t)$ , który jest nieskończenie wysoki, nieskończenie krótki i ma pole równe 1. W warunkach rzeczywistych i symulacyjnych stosuje się różne przybliżenia idealnego impulsu. Poprawna realizacja tego badania wymaga, aby **impuls na wejściu następował zawsze w warunkach stanu ustalonego**.



Rys. I-20. Badanie reakcji na wymuszenie impulsowe

### 3.1.3. Badania obiektu w dziedzinie częstotliwości (odpowiedzi częstotliwościowe)

Charakterystyki częstotliwościowe opisują odpowiedź obiektu na wymuszenia sinusoidalne. Jeśli na wejście obiektu liniowego zostanie podany przebieg sinusoidalny, to na wyjściu obiektu w stanie ustalonym (Rys. I-21) będzie przebieg sinusoidalny o takiej samej częstotliwości (pulsacji), ale wzmacniony ( $x_m/u_m$ ) i przesunięty w fazie ( $\varphi$ ). Własności dynamiczne obiektu sprawiają, że to wzmacnienie i przesunięcie fazowe zależy od częstotliwości (pulsacji). Zależności te są przedstawiane na różnego typu charakterystykach częstotliwościowych obiektu, wynikających z różnych form transmitancji widmowej:



Rys. I-21. Badanie reakcji na wymuszenie sinusoidalne

$$G(s) = G(j\omega) = P(\omega) + jQ(\omega) = A(\omega)e^{j\varphi(\omega)} \quad (I-16)$$

Najłatwiej jest wygenerować charakterystyki częstotliwościowe symulacyjnie dla modeli liniowych (równań stanu, transmitancji) za pomocą specjalistycznej funkcji, analogicznie jak w przypadku odpowiedzi skokowej (p. 2.4.2÷2.4.5):

	Zależność	Matlab/Control	Scilab
Charakterystyka Nyquista (ch. amplitudowo-fazowa)	$P(Q)$	nyquist(obiekt)	nyquist(objekt)
Charakterystyki Bodego (log.ch. modułu i fazy)	$M(\omega) = 20 \lg A(\omega)$ $\varphi(\omega)$	bode(objekt)	bode(objekt)

W rzeczywistych warunkach zdjęcie charakterystyki częstotliwościowej obiektu nie zawsze jest realne, wymaga bowiem aby była możliwość wygenerowania na wejściu przebiegów sinusoidalnych o różnych częstotliwościach (pulsacjach). Jest też pracochłonne, ponieważ po ustaleniu wymuszenia konieczne jest osiągnięcie stanu równowagi, a eksperyment należy powtórzyć dla różnych częstotliwości.

**Dodać: Opis osi, skale, ... Zakresy częstotliwości (pulsacji). Rysowanie na podstawie asymptot**

## 3.2. Podstawowe parametry (wskaźniki) układów dynamiki

### 3.2.1. Definicja modeli

W punkcie 2.4 przedstawiono podstawowe sposoby definiowania modeli w trybie tekstowym, na podstawie macierzy równań stanu oraz na podstawie współczynników licznika i mianownika transmitancji (w Matlabie funkcje `ss`, `tf`, a w Scilabie funkcja `syslin`):

	Matlab/Control	Scilab
Równania stanu	obS = ss(A, B, C, D);	obS = syslin('c',A,B,C,D);
Transmitancje	obT = tf([b1 b0], [a2 a1 a0]);	obT = syslin('c',b1*s+b0, a2*s^2+a1*s+a0);
	obT = (b1*s+b0) / (a2*s^2+a1*s+a0);	obT = (b1*s+b0) / (a2*s^2+a1*s+a0);

Definicję złożonych transmitancji znacznie ułatwia zastosowanie wyrażeń ze zmienną  $s$ . Wykorzystując zmienną  $s$  można w prosty sposób definiować transmitancje podstawowych członów (obiektów) dynamiki:

	Matlab/Control	Scilab
Inercja 1 rzędu	G1 = k / (T*s+1);	G1 = k / (T*s+1);
Wzmocnienie	Gk = k;	
Różniczkowanie	Gd = Td*s;	
Opóźnienie To	Go = exp(-To*s);	aproxymacja Pade?

W Matlabie dostępna jest jeszcze jedna funkcja do definiowania transmitancji na podstawie wartości wzmocnienia, biegunów i zer transmitancji - `zpk()`.

**Modele złożone**

### 3.2.2. Konwersja modeli

Modele zdefiniowane w postaci równań stanu czy transmitancji można przekonwertować do innej postaci za pomocą dedykowanych funkcji:

Konwersja	Matlab/Control	Scilab
- równań stanu	model_tf = ss2tf(model_ss);	model_tf = ss2tf(model_ss);
- transmitancji	model_ss = tf2ss(model_tf); model_zpk = tf2zpk(model_tf);	model_ss = tf2ss(model_tf);

- transmitancji	<code>model_ss = zp2ss(model_zpk);</code> <code>model_tf = zp2tf(model_zpk);</code>	
-----------------	--	--

W Matlabie możliwe jest też zastosowanie każdej z funkcji ss, tf, zpk nie tylko do definiowania modeli, ale także do konwersji modelu z jednej postaci na inną – konwersja nastąpi gdy parametr funkcji będzie nazwa obiektu LTI.

### 3.2.3. Odczytywanie parametrów

- funkcje

- na wykresach

## 3.3. Zastosowanie narzędzi wspomagających – Matlab/Linear Analysis

### 3.3.1. Interfejs Linear Analysis dla obiektów LTI

### 3.3.2. Interfejs Linear Analysis dla schematów



## 4. Przypadki szczególne

Czy to potrzebne – może przenieść do PPS1

### 4.1. Obiekty niestabilne

### 4.2. Obiekty astatyczne

Występuje całkowanie, Układ na granicy stabilności, Brak punktu równowagi

„lewe” implementacje

- zapętlone niby transmitancje

– może uwaga w rozdziale z blokami Transfer Fcn?

- albo zmienić 2.5 na inne metody – poprawne i niepoprawne (nietypowe)

skalowanie???

Dodać - potwierdzenie symulacji na podstawie parametrów, np. biegunów