AKADEMIA MORSKA W GDYNI

Wydział Elektryczny

Katedra Automatyki Okrętowej

Nr ewidencyjny.....

Data wydania tematu.....

Data złożenia pracy.....

PRACA DYPLOMOWA
MAGISTERSKA

Dyplomant:	**Daniel Czarkowski**	**Nr albumu: 10352/E**
Specjalność:	**Elektroautomatyka Okrętowa**	
Promotor:	**Dr hab. inż. prof. ndzw. Roman Śmierzchalski**	Ocena:
Recenzent:	**Dr inż. Mirosław Tomera**	Ocena:
Egzamin dyplomowy:	**data: 12.07.2002r.**	Ocena:

TEMAT:Identyfikacja oraz strojenie regulatora PID przy wykorzystaniu pakietu MATLAB

Gdynia 2002

SPIS TREŚCI

SPIS RYSUNKÓW I TABEL

Wstęp

Z wykorzystaniem automatyki spotykamy się na każdym kroku, nawet wtedy, gdy sobie tego do końca nie uświadamiamy. Weźmy tak prozaiczną sprawę, jak kupno bananów. Aby znalazły się w Polsce trzeba je przetransportować z krajów tropikalnych. W tym celu należy utrzymać stałą temperaturę w chłodni, co jest realizowane przez zastosowanie regulatora PID. Odpowiedni dobór nastaw regulatora pozwala na bezpieczne przewiezienie wspomnianych bananów do portu przeznaczenia. Już ten przykład ilustruje, jak ważna jest wiedza o strojeniu regulatorów, wiedza zdobywana na uczelni. Niestety praktyka wskazuje, iż mimo istnienia 181 naukowych metod doboru nastaw, nadal dominuje sposób empiryczny, o czym przekonałem się podczas pracy w firmie Thoresen & Moen.

Jedną z wyżej wspomnianych metod naukowych są algorytmy genetyczne, które zostały zaimplementowane do mojego programu. Naśladują one procesy zachodzące w ewolucji. Algorytmy te, używając prostych metod kodowania i mechanizmów reprodukcji, pozwalają na rozwiązywanie dość złożonych problemów optymalizacyjnych. Metoda ta zalicza się do elementów sztucznej inteligencji (*ang. Artificial Intelligence, AI*), tak jak sieci neuronowe, systemy ekspertowe i obliczenia rozmyte. W ostatnich latach wzrost wydajności komputerów pozwolił na bardzo szybki rozwój tych technik i zastosowanie ich w wielu dziedzinach życia.

Celem niniejszej pracy jest strojenie regulatora PID metodą ewolucyjną. Dodatkowo praca jest poszerzona o identyfikację obiektu. Pierwszym krokiem strojenia regulatora jest otrzymanie matematycznego modelu obiektu inercyjnego pierwszego rzędu, jakim jest zbiornik z cieczą. Jest to osiągnięte przez podanie skoku jednostkowego na badany obiekt. Dane są przesyłane do Matlaba za pomocą *Data Acqustion Unit*. Program ma na celu zniwelowanie zakłóceń poprzez usuniecie szumu wysokiej częstotliwości. W programie zastosowano trzy metody identyfikacyjne, dwie z nich są dostępne w *Identification Toolbox*, natomiast metoda klasyczna została zaimplementowana przeze mnie. Po otrzymaniu matematycznego modelu, program znajduje nastawy regulatora PI(D) za pomocą dwóch technik optymalizacyjnych, jakimi są algorytmy genetyczne oraz komputerowe wspomaganie projektowania systemów nieliniowych (*ang. **N**onlinear **C**ontrol **D**esign Blockset Toolbox, NCD*) dostarczony przez *Mathworks*. Obsługę programu ułatwia Graficzny Interfejs Użytkownika (*ang. **G**rafical **U**ser **I**nterface)*.

W celu uniknięcia niejasności w pracy stosuje się angielską nomenklaturę.

W rozdziale pierwszym przedstawiono problem obiektu regulowanego, czyli skupiono się na identyfikacji obiektu. Badania identyfikacyjne oraz zastosowanie różnych metod identyfikacyjnych zostały zamieszczone w rozdziale 5.1.

W rozdziale drugim przedstawiono rodzaje układów regulacji, sklasyfikowano układy automatyki oraz podział regulatorów.

W rozdziale trzecim przedstawiono strukturę regulatora PID oraz zawarto informacje na temat regulatora cyfrowego.

W rozdziale czwartym przeanalizowano trzy metody strojenia regulatora. Począwszy od algorytmów genetycznych przez oprogramowanie optymalizacyjne dostarczane przez *Mathworks* i skończywszy na coraz mniej popularnej metodzie Zieglera – Nicholsa. Zamieszczono również podstawy działania algorytmów genetycznych.

W rozdziale piątym zamieszczono badania programu, który identyfikuje obiekt oraz poszukuje nastaw regulatora używając metod opisanych w rozdziale czwartym. Skupiono się przede wszystkim na przeanalizowaniu różnych obiektów regulacji dla algorytmów genetycznych.

1 Identyfikacja obiektu sterowania w zamkniętym układzie regulacji

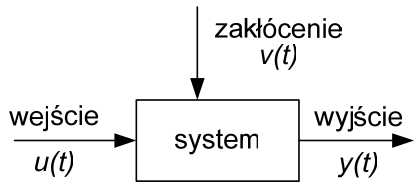
W celu znalezienia nastaw regulatora PID należy zidentyfikować obiekt podlegający sterowaniu, czyli wyznaczyć jego model matematyczny. Definiuje on w sposób ścisły zachowanie się obiektu w określonych warunkach, które są opisane przez wejścia i wyjścia z obiektu w chwili obecnej i w przeszłości. Z natury rzeczy (nieskończona ilość wielkości oddziałujących na zachowanie się obiektu, nieliniowości, trudność ścisłego zdefiniowania wielkości wejściowych i wyjściowych) model jest pewnym przybliżeniem rzeczywistego obiektu, idealizacją z ograniczającymi założeniami (ograniczona ilość wejść i wyjść, liniowość). Stosuje się różne postacie modeli w zależności od przeznaczenia tworzonego modelu i struktury identyfikowanego obiektu. Modele opisujące rzeczywiste obiekty, są podzielone na modele dynamiczne i liniowe, wielowejściowe i wielowyjściowe (*ang. Multiple Input Multiple Output*, MIMO) oraz modele z jednym wejściem i jednym wyjściem (*ang*. *Single Input Single Output*, SISO).[1]

Identyfikacja jest przeprowadzana na podstawie informacji pomiarowej o wielkościach wejściowych i wyjściowych obiektu. Ta informacja pomiarowa we współczesnych komputerowych systemach identyfikacji to próbki sygnałów wejściowych i wyjściowych. W odniesieniu do sygnałów również stosuje się często idealizację, przybliżając rzeczywiste sygnały ich idealnymi odpowiednikami opisywanymi matematycznie. Ten zabieg pozwala na łatwiejszą ich analizę w zadaniach identyfikacji. Niektóre sygnały idealne nie występują w rzeczywistości z powodu ograniczeń zmian energii w czasie (impuls Diraca, skok jednostkowy), są jednak, przy spełnieniu określonych założeń, dobrym przybliżeniem sygnałów rzeczywistych.[2]

Wstępnym etapem identyfikacji jest określenie charakteru obiektu na podstawie rejestracji jego sygnałów (statyczny czy dynamiczny, jeśli dynamiczny to jakiego rzędu, o jakiej dynamice). Dlatego podstawową wiedzą w identyfikacji obiektów jest znajomość zachowania się standardowych obiektów (pierwszego i drugiego rzędu) pod wpływem standardowych sygnałów pobudzających (np. skok, impuls jednostkowy, pobudzenie sinusoidalne).

[1] Więcej na temat elementów automatyki w rozdziale 2.5

[2] Tomasz Twardowski, *Laboratorium komputerowej identyfikacji obiektów*, Skrypt Akademii Górniczo – Hutniczej, Zakład Metrologii, Kraków 2001 r.



Rys. 1.1 Przykład układu dynamicznego

System dynamiczny może być schematycznie przedstawiony tak jak na powyższym rysunku. Zachowanie układu wyznaczają sygnały wejściowe *u(t)* i zakłócenia *v(t)*, przy czym użytkownik może wpływać tylko na *u(t)*. W niektórych zastosowaniach sygnały wejściowe mogą nie występować. Istotnych informacji o układzie dostarcza sygnał wyjściowy.[3]

1.1 Modele matematyczne liniowych systemów dynamicznych

Liniowy model dynamiczny to model spełniający zasadę superpozycji, tzn. model, którego działanie opisuje operator *F* spełnione jest równanie:

$$F(a \cdot u_1(t) + b \cdot u_2(t)) = a \cdot F(u_1(t)) + b \cdot F(u_2(t))$$

Jak łatwo sprawdzić operatorem liniowym jest zarówno operator całkowania jak i operator różniczkowania.

1.1.1 Postacie matematycznych modeli liniowych systemów dynamicznych

Liniowe systemy dynamiczne możemy opisać za pomocą czterech postaci modeli matematycznych:

- równanie różniczkowe:

$$y(t) + a_1 \frac{dy(t)}{dt} + \ldots + a_n \frac{d^n y(t)}{dt^n} = b_0 u(t) + b_1 \frac{du(t)}{dt} + \ldots + b_m \frac{d^m u(t)}{dt^m}$$

Systemy fizyczne spełniają warunek realizowalności: $m \le n$.

[3] Torsten Söderström, Pstre Stoica, *System Identification*, Prentice Hall International 1994r.

Pojedyncze równanie opisuje układ SISO (jedno wejście, jedno wyjście) z określonymi warunkami początkowymi.

- równania stanu

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

A – macierz kwadratowa $n \times n$, **B** – macierz $n \times m$,

C – macierz $p \times n$, **D** – macierz $p \times m$,

n – ilość stanów, m – ilość wejść, p – ilość wyjść.

Macierzowe równanie stanu opisuje układ MIMO (wiele wejść, wiele wyjść).

Ogólne rozwiązanie równań stanu ma postać:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau$$

Jak wynika z powyższego równania zachowanie się obiektu nie zależy od historii sygnałów wejściowych, co osiągnięto dzięki wprowadzeniu do modelu zmiennych stanu.

- transmitancją operatorową *G(S)* elementu lub układu nazywa się stosunek transformaty wielkości wyjściowej y(s) do transformaty wielkości wyjściowej *x(s)* przy założonych warunkach początkowych.[4] Jedna transmitancja opisuje układ SISO.

$$G(s) = \frac{y(s)}{u(s)} = \frac{£\{y(t)\}}{£\{u(t)\}} = \frac{b_0 + b_1 s + \ldots + b_m s^m}{1 + a_1 s + \ldots + a_n s^n}$$

Ogólne wyrażenie opisujące odpowiedź na wejście $u(t)$ ma postać:

$$y(t) = £^{-1}\{G(s)£\{u(t)\}\}$$

Podstawiając $s=j\omega$ otrzymujemy transmitancję widmową. W szczególnym przypadku pobudzenia harmonicznego o określonej pulsacji ω odpowiedź ustalona na wejściowy sygnał sinusoidalny ma postać sinusoidalną o amplitudzie i fazie zmienionej przez transmitancję w punkcie $j\omega$, tj.:

$$Y(j\omega) = U(j\omega) \cdot G(j\omega)$$

[4] zob. rys 2.4

- odpowiedź impulsowa

$$g(t) = F(\delta(t))$$

Impuls Diraca $\delta(t) \neq 0$ tylko dla t=0 oraz $\int_{-\infty}^{+\infty} \delta(t)dt = 1$

Ogólne wyrażenie opisujące odpowiedź na wejście $u(t)$ ma postać splotową:

$$y(t_0) = g(t) * u(t)\big|_{t=t0} = \int_0^{\infty} g(\tau)u(t_0 - \tau)d\tau$$

1.1.2 Związki między poszczególnymi postaciami modeli.

W automatyce występują następujące relacje pomiędzy poszczególnymi postaciami modeli:

- równanie różniczkowe $\Rightarrow$ równania stanu. W celu uzyskania tego przekształcenia należy skorzystać z metody podstawienia zmiennej stanu za pochodną.
- równanie różniczkowe $\Rightarrow$ transmitancja operatorowa
  transformata Laplace'a
- równania stanu $\Rightarrow$ transmitancja operatorowa

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{1} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

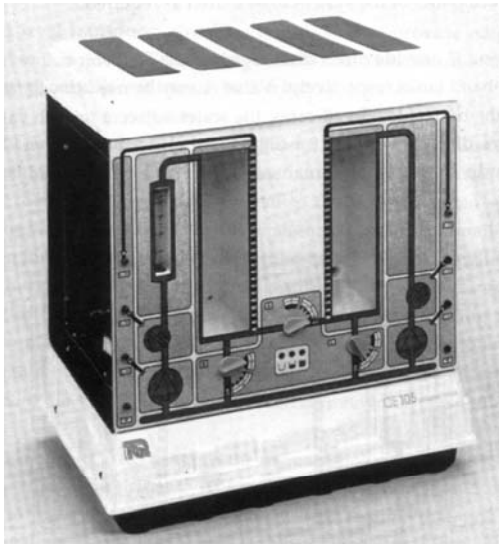
- odpowiedź impulsowa $\Rightarrow$ transmitancja operatorowa

$$G(s) = £(g(t))$$

1.2 Obiekt identyfikowany

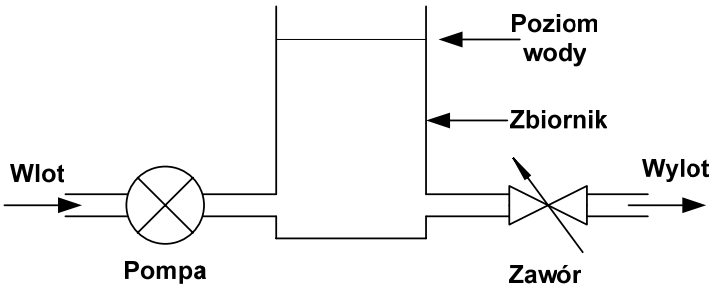
Jako przykład identyfikacji rozważano zbiornik z cieczą **CE105 Coupled Tanks Apparatus**[5], który jest pokazany na poniższym rysunku. W rozdziale 5.1 przeprowadzono identyfikację tego obiektu – inercyjny pierwszego rzędu bez opóźnienia transportowego.

[5] oficjalna strona producenta *Tecquipment Limited* http://www.tq.com/index.html



Rys. 1.2 Zbiornik

Poniższy schemat ilustruje przypływ cieczy przez zbiornik. Pompa jest używana do napełniania zbiornika przy stałej wydajności. Na wylocie zbiornika jest umieszczony regulowany zawór. Zawór ten został tak ustawiony, aby pompa pracująca ze stałą wydajnością napełniła zbiornik do pewnego ustalonego poziomu, czyli ciecz znajdująca się w zbiorniku utrzymywała się na stałym poziomie. Gdy warunki te zostaną osiągnięte, to taki stan jest nazywany równowagą (*łac. Equilibrium*)



Rys. 1.3 Schemat zbiornika

Co się stanie w przypadku rozregulowania zaworu wylotowego? Program został zaadaptowany do tego typu sytuacji. Założono w nim, że pompa przestanie pracować, gdy przez okres co najmniej 120 sekund będzie się utrzymywał stały poziom cieczy w zbiorniku. Innymi słowy proces identyfikacji zostanie zakończony, gdy *s(n)=b* oraz *s(n+120)=b,* wtedy pompa przestanie pracować. Kod programu został zamieszczony w pliku `calc_rt.m`[6].

[6] patrz załącznik A

2 Człony funkcjonalne układów automatyki

2.1 Regulatory

Regulator jest to urządzenie, które porównuje sygnał przychodzący z organu pomiarowego z sygnałem wartości zadanej i w zależności od tej różnicy działa na urządzenie wykonawcze w takim kierunku, aby tę różnicę zmniejszyć.[7]

Regulatory podobnie jak większość układów automatyki można sklasyfikować pod różnymi względami:

- ze względu na wykorzystanie energii pomocniczej można podzielić je na:
  - regulatory bezpośredniego działania tzn. nie korzystające ze źródeł energii pomocniczej przy przedstawianiu elementu wykonawczego,
  - regulatory pośredniego działania tzn. korzystające z energii pomocniczej przy przedstawianiu elementu wykonawczego,
- ze względu na rodzaj nośnika energii wyróżnia się:
  - regulatory elektryczne,
  - regulatory pneumatyczne,
  - regulatory hydrauliczne,
  - regulatory mieszane (elektropneumatyczna, elektrohydrauliczne).
- ze względu na rodzaj sygnału wyjściowego regulatory mogą być:
  - ciągłe – to regulatory, które spełniają prawo regulacji typu P (regulator proporcjonalny), I (regulator – całkujący), PI (regulator proporcjonalno – całkujący), PD (regulator proporcjonalno – różniczkujący)[8],
  - nieciągłe – do klasy regulatorów nieciągłych mażemy zaliczyć regulatory przekaźnikowe dwupołożeniowe, trójpołożeniowe oraz krokowe czyli regulatory trójstanowe sprzężone z silnikiem krokowym.

Regulatory pracujące w układzie automatycznej regulacji nie mogące sprowadzić uchybu ustalonego do zera nazywamy regulatorami statycznymi. Należą do nich regulatory typu P oraz regulatory typu PD. Natomiast regulatory astatyczne to takie, które sprowadzają uchyb ustalony do zera. Do tej grupy należą regulatory typu I, PI, PID.

---

[7] W. Chotkowski, *Podstawy automatyki,* Skrypt Politechniki Gdańskiej

[8] więcej na temat regulatorów PID w następnym rozdziale

Klasyfikacji regulatorów dokonywać możemy również z punktu widzenia zmian wartości zadaniowej:

- regulatory stałowartościowe – są to regulatory majce utrzymać stałą wartość zadaną np. temperaturę, ciśnienie, poziom cieczy,
- regulatory programowe – są to takie regulatory, które potrafią śledzić wartość zadaną, której zmiany zostały wcześniej zaprogramowane,
- regulatory nadążne – są to regulatory pracujące w układzie regulacji nadążnej czyli takich, w których wartość zadana jest zmienna ale zmiany nie są wcześniej znane,
- regulatory ekstremalne – są to regulatory mające za zadanie utrzymanie ekstremalnego punktu pracy układu znajdującego się na charakterystyce statycznej lub dynamicznej obiektu,
- regulatory adaptacyjne – są to regulatory, których własności dynamiczne są optymalizowane przez automatyczne dostosowanie się warunków pracy obiektu. Adaptacja może dotyczyć nastrajania parametrów regulatora (adaptacja parametryczna) lub nastrajania struktury (adaptacja strukturalna). Przykładem technicznego rozwiązania są autopiloty okrętowe, zawierające tor adaptacji do zmiennych parametrów statku jako obiektu sterowania,
- regulatory optymalne – spełniają określony wskaźnik jakości, ujmujący globalnie własności dynamiczne obiektu i mający określone uzasadnienie ekonomiczne lub energetyczne. Techniczna realizacja tych regulatorów jest bardzo trudna, dlatego też buduje się regulatory suboptymalne (prawie optymalne), tzn. takie, które z pewnym przybliżeniem realizują założony wskaźnik jakości.

2.2 Sterowanie, obiekt i proces sterowania

Automatyka jest dziedziną wiedzy zajmującą się zagadnieniami automatycznego sterowania procesów. Zawiera ona teorię sterowania automatycznego i automatyzację, czyli wprowadzenie urządzeń realizujących to sterowanie.[9]

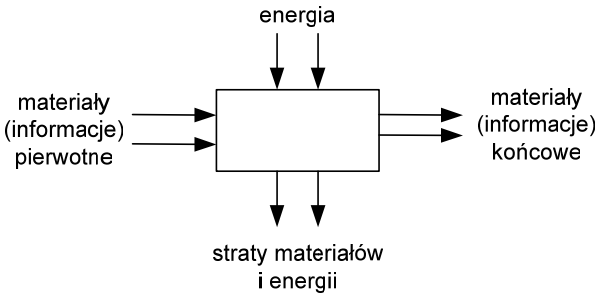
Zasięg automatyzacji obejmuje procesy technologiczne, wśród których można wymienić następujące zasadnicze grupy:

- przetwórcze (zmiana stanu fizycznego lub składu chemicznego materiałów),

[9] Marek Żelazny, *Podstawy automatyki,* s.13, PWN, Warszawa 1976r.

- obróbcze (zmiana kształtu),
- transportowe.

Ogólnie, proces technologiczny polega na wzajemnym oddziaływaniu wielu strumieni materiałów i energii.



Rys. 2.1 Ogólne przedstawienie procesu technologicznego

Sterowanie procesu – polega na oddziaływaniu na strumienie energii lub materiałów w taki sposób, aby zrealizowany został zamierzony przebieg procesu. Oddziaływanie to może być prowadzone przez człowieka (sterowanie ręczne) lub przez zespół środków technicznych zastępujących człowieka w czynnościach nadzoru i wpływania na przebieg procesu (sterowanie automatyczne).

Urządzenia sterujące – jest to urządzenie, które kieruje działaniem całego procesu lub obiektu sterowania. Wygenerowane impulsy w układzie sterującym powodują wykonanie poszczególnych rozkazów, zapewniając przy tym odpowiednią synchronizację poszczególnych faz ich wykonania.

Obiekt i proces sterowania. Pojęcie „obiekt regulacji" (sterowania) używane jest w dwojakim sensie. W przypadku, gdy mówimy o własnościach statycznych i dynamicznych, obiekt regulacji należy rozumieć jako jeden z elementów układu, mający wielkość wejściową i wyjściową, określony równaniem różniczkowym, transmitancją operatorową i współrzędnymi stanu. Obiektem jest wówczas proces, którego przebieg polega na regulacji sterowania, np. proces zmiany poziomu wody w kotle.

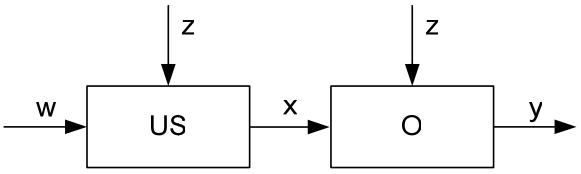
W drugim przypadku pojęcie obiektu ma sens aparaturowy. Oznacza ono wówczas aparaturę technologiczną, w której zachodzi proces regulowany lub sterowany, np. kocioł. Dla uniknięcia nieporozumień i rozróżnienia obu znaczeń, w tym przypadku używane jest pojęcie „obiekt automatyzowany".

2.3 Sterowanie w układzie otwartym i zamknięty

Układ automatyki jest to zespół elementów biorących bezpośredni udział w sterowaniu automatycznym danego procesu oraz elementów pomocniczych, uporządkowany na zasadzie ich wzajemnej współpracy, tzn. zgodnie z kierunkiem przekazywania sygnałów. Można wyróżnić dwa podstawowe rodzaje sterowania automatycznego:[10]

- sterowanie w układzie otwartym,
- sterowanie w układzie zamkniętym.

Układ otwarty – jest to układ, w którym nie ma oddziaływania wielkości wyjściowej na wielkość wyjściową. Układy otwarte nie są w stanie równoważyć zmian wewnętrznych własności obiektów sterowania oraz z zasady nie mogą sterować obiektami niestabilnymi.

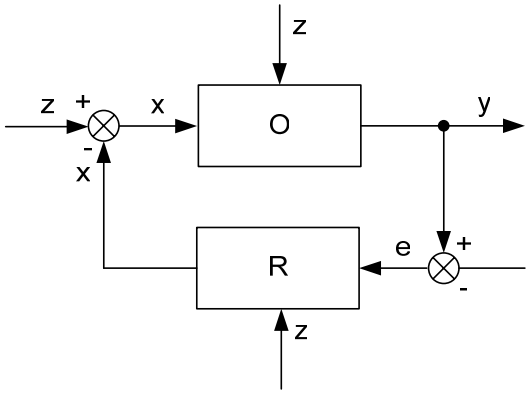


Rys. 2.2 Schemat otwartego układu sterowania

gdzie: *US* – urządzenie sterujące; *O* – Obiekt sterowany; w – sygnał wymuszenia (wartość zadania wielkości sterowanej); *x* – sygnał nastawiający; *y* – wielkość sterowana; z– sygnały zakłócające (zakłócenia).

Układ zamknięty – jest to układ, w którym istnieje sprężenie zwrotne. Jednym z podstawowych układów automatyki jest układ z ujemnym sprężeniem zwrotnym. Układ ten charakteryzuje się mniejszym wpływem zakłóceń działających na układ regulacji, mniejszymi, co do liniowości elementów w porównaniu z układem otwartym.

[10] Marek Żelazny, *Podstawy automatyki,*s.16, PWN, Warszawa 1976 r.



Rys. 2.3 Schemat zamkniętego układu regulacji

gdzie: *O* – obiekt regulacji (sterowania); *R* – regulator (urządzenie sterujące), *w* – wartość zadania wielkości regulowanej (sterowanej); *e* – odchylenie regulacji (sterowania); *x* – sygnał nastawiający; *y* – wielkość regulowana (sterowana); z – zakłócenia;

Tor główny – występuje w układzie otwartym jak i zamkniętym. W torze głównym znajduje się zawsze obiekt regulacji (sterowania). Odpowiada on podstawowemu strumieniowi materiału (lub energii) w danym procesie i jest zwykle miejscem występowania najistotniejszych dla procesu zakłóceń.

Tor sprężania zwrotnego – jest to tor, w którym znajdują się elementy mierzące wielkość regulowaną (sterowaną), porównując je z wartością zadaną.

Węzeł sumacyjny – element schematu blokowego, reprezentujący operację matematyczną dodawania lub odejmowania. Wielkość wyjściowa węzła symulacyjnego jest sumą lub różnicą wielkości wejściowych. Węzeł sumacyjny jest najprostszym elementem o dwóch wejściach i jednym wyjściu.

Węzeł zaczepowy – jest to punkt rozgałęzienia się dróg sygnału w schemacie blokowym. Wszystkie wielkości wychodzące z węzła zaczepowego są równe.

2.4 Rodzaje układów automatyki

2.4.1 Podział ze względu na zadanie układów regulacji

Układy stabilizujące – tzn. układy regulacji stałowartościowej, w których wartość zadania jest stała. Zadaniem układu jest utrzymanie stałej wartości wielkości regulowanej, mimo działających zakłóceń.

Układy programowe – tzn. układy regulacji programowej i układy sterowania programowego, w których wartość zadania jest z góry określoną funkcją czasu, *w= w(t)*. Jako przykład może tu posłużyć układ programowej regulacji temperatury w piecu hartowniczym.

Układy nadążne (śledzące) – zwane też serwomechanizmami, są to zamknięte układy sterowania, w których wartość zadana jest nieznaną funkcją czasu. Zadanie układu jest spowodowanie nadążeniem wielkości sterowanej za zmianami wartości zadanej. Przykładem układu nadążnego może być autokompenastor, tzn. rejestrator z zamkniętym układem sterowania położenia pisaka.

Układ sterowania optymalnego – zadanie układu polega na maksymalizacji lub minimalizacji funkcji wielu zmiennych $f(x_1, ... ,x_n)$, która ma zwykle taki sens techniczny lub ekonomiczny jak wydajność produkcji, zysk, koszt produkcji, zużycie paliwa itp. Funkcja ta nazywana jest wskaźnikiem jakości, a jej minimalizację lub maksymalizację przeprowadza się z uwzględnieniem ograniczeń na wartości zmiennych $x_1, ... x_n$

Układy sterowania sekwencyjnego – zadaniem tych układów jest zapewnienie wykonania składowych operacji procesu technologicznego w określonej kolejności (sekwencji). Sterowanie daje się wówczas sprowadzić do odpowiednio uwarunkowanego załączenia lub wyłączenia poszczególnych urządzeń i realizowane jest przez układy przełączające, w których rolę regulatora spełnia tak zwany układ logiczny.

Układu regulacji ekstremalnej – są wykorzystywane wówczas, gdy charakterystyka statyczna jest krzywą ekstremalną (posiada maksimum lub minimum). Zadaniem układu regulacji ekstremalnej jest utrzymywanie wielkości regulowanej stale na wartości ekstremalnej w danych warunkach.

Układy adaptacyjne – układ sterowania automatycznego mający własności przystosowania się do zmiennych warunków pracy całego układu lub jego części, w celu utrzymania pożądanego działania lub stanu układu. W układach tych regulatory nastraja się automatycznie w zależności od zmiennych warunków pracy, tak aby spełnić określone kryterium jakości. Przystosowanie może nastąpić przez zmianę wartości zadanych (np. punktu pracy), zmianę parametrów lub struktury układu. Układy adaptacyjne wymagają ciągłego badania procesu regulacji dla uzyskania aktualnych informacji o charakterystyce obiektu i zakłóceń.[11]

[11] S. Perycz, *Podstawy automatyki,* Skrypt dla instytutu okrętowego Politechniki Gdańskiej, Gdańsk 1980r.

2.4.2 Podział ze względu na sposób działania elementów układów regulacji

Układy o działaniu ciągłym – wszystkie elementy układu działają w sposób ciągły w czasie i poziomie, tzn. wszystkie sygnały są funkcjami ciągłymi i mogą przybierać każdą wartość (od najmniejszej do największej) znajdującą się w normalnym obszarze ich zmienności.

Układy o działaniu dyskretnym – przynajmniej jeden element układu działa w sposób dyskretny, tzn. jego sygnały mogą przyjmować tylko niektóre, wybrane wartości lub występują tylko w niektórych, wybranych chwilach czasu. W tej grupie wyróżnia się:

- układy przekaźnikowe – sygnały na wyjściu elementów przekaźnikowych mogą przyjmować najczęściej tylko dwie lub trzy wartości (np. zero i maksimum, obwód rozwarty lub zwarty), przy czym przejście od jednej do drugiej wartości następuje wtedy, gdy sygnał wyjściowy przekroczy punkt (strefę) przełączenia,
- układy impulsowe – sygnały na wyjściu elementów impulsowych pojawiają się tylko w chwilach impulsowania, np. co 10 lub 100 sekund, przy czym szerokość (czas trwania) impulsu może być stała, a wysokość (amplituda) zmienna lub odwrotnie. Zależnie od tego, czy nośnikiem informacji jest wysokość, czy szerokość sygnału, mówi się o układach z modulacją wysokości (amplitudy) lub o układach z modulacją częstotliwości sygnału, kiedy zmienny jest okres impulsowania i on właśnie jest nośnikiem informacji.

2.4.3 Podział ze względu na modelowany typ elementów układów regulacji

Układy liniowe – układy, które zawierają wyłącznie elementy liniowe, tzn. elementy o prostoliniowych charakterystykach statycznych, opisywane za pomocą liniowych równań różniczkowych, całkowych lub algebraicznych. Elementy i układy liniowe spełniają zasadę superpozycji.

Układy nieliniowe – układy zawierające przynajmniej jeden element nieliniowy. Element ten, a zatem i cały układ nie spełnia zasady superpozycji i nie może być opisany za pomocą równania liniowego.

2.4.4 Inne podziały klasyfikacyjne

Układy analogowe – w układach tych wielkość regulowana y jest mierzona i przetwarzana na inną, której przebieg jest pewną analogią przebiegu y. Wyniki pomiaru mogą być wyrażone w postaci: ciśnienia, napięcia, natężenia prądu, przesunięcia lub innej wielkości fizycznej będącej sygnałem analogowym ciągłym, Dalsze operacje w układzie, jak porównywanie zmierzonej wartości wielkości regulowanej z jej wartością zadaną, wytwarzania sygnału wyjściowego regulatora, dokonywane są również na sygnałach analogowych ciągłych.[12]

Układy cyfrowe – w układach tych wynik pomiaru jest przetwarzany na sygnał cyfrowy i wyrażany w postaci liczby y_1. Wartość zadana jest również wyrażana w postaci liczby W_1. Obliczanie odchylenia regulacji e_1 i sygnału wyjściowego X_1 regulatora ma charakter operacji cyfrowych, dopiero przed elementem wykonawczym konieczne jest przejście na analogowy sygnał nastawiający X.[13]

Układu specjalne (tzw. inteligentne) – w tej grupie wyróżnia się:

- Układy ekspertowe – są to programy komputerowe, które na podstawie zgromadzonych w pamięci danych mogą wyciągać wnioski i podejmować decyzje, działając w sposób zbliżony do rozumowania człowieka.[14]
- Sieci neuronowe – można traktować jako nowoczesne systemy obliczeniowe, które przetwarzają informacje wzorując się na zjawiskach zachodzących w mózgu człowieka. Największą zaletą sieci neuronowych jest możliwość ich uczenia i adaptacji.[15]
- Algorytmy genetyczne – naśladują procesy zachodzące w świecie organizmów żywych, a mianowicie ewolucji i związanej z nią naturalnej selekcji występującej w populacjach żywych osobników. Algorytmy genetyczne stanowią wzorowaną na naturalnej ewolucji metodą rozwiązywania problemów, głównie zagadnień optymalizacyjnych.[16]
- Zbiory rozmyte – w układach tych nie jest wymagana wiedza szczegółowa, opisująca w sposób matematyczny zależność funkcyjną między wyjściem, a wejściem

[12] zob. rys 3.1a

[13] zob. rys 3.1b

[14] więcej J. Mulawka, *Systemy Ekspertowe,* WNT, Warszawa 1996r.

[15] D. Rutkowska, M. Piliński, L. Rutkowski, *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte,* PWN, Warszawa 1999r.

[16] zasada działania algorytmów genetycznych została opisana w rozdziale 4.2

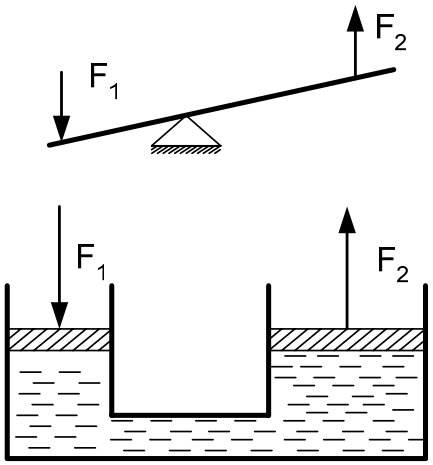
sterownika. W odróżnieniu od zwykłych sterowników posługujemy się wiedzą jakościową „jak", a nie ilościową „ile". Decyzje są podejmowane przez układ w oparciu o bazę reguł zapisanych w postaci implikacji IF – THEN. Największą wadą tych układów jest brak możliwości uczenia się i adaptacji.

2.5 Elementy automatyki

Element automatyki jest to dowolny podzespół, zespół, przyrząd lub urządzenie występujące w układach automatyki, w którym wyróżnić można sygnał wejściowy i wyjściowy. W najprostszym przypadku elementy mają tylko jeden sygnał wejściowy i jeden wyjściowy (*ang. Single Input Single Output*, SISO).[17]

Rys. 2.4 Umowne oznaczenie elementu automatyki

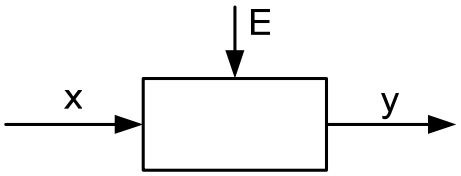
Przykładem elementu automatyki o jednym wejściu i jednym wyjściu może być dźwignia, w której sygnałem wejściowym jest siła F_1 działająca (w dół) na jeden z jej końców, a sygnałem wyjściowym jest siła skierowana do góry F_2 równa sile F_1.



Rys. 2.5 Element automatyki o jednym wejściu i jednym wyjściu

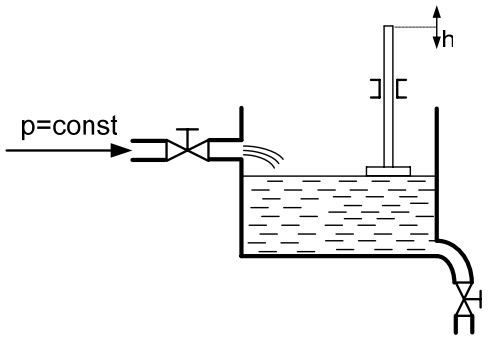
Element wzmacniający – to taki element, w którym energia otrzymywana na wejściu jest większa od energii doprowadzonej do wejścia, możliwe jest to do osiągnięcia po dostarczeniu dodatkowej energii z zewnątrz.

[17] Marek Żelazny, *Podstawy automatyki,* PWN, Warszawa 1976



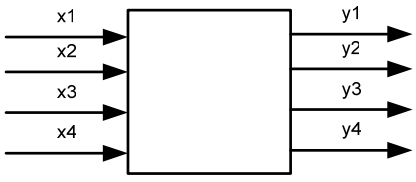
Rys. 2.6 Zasada działania elementu wzmacniającego

Przykładem elementu wzmacniającego może być zbiornik, w którym sygnałem wejściowym jest kąt obrotu zaworu, przez który dostarczana jest ciecz do zbiornika. Sygnałem wyjściowym natomiast przesunięcie pływaka.



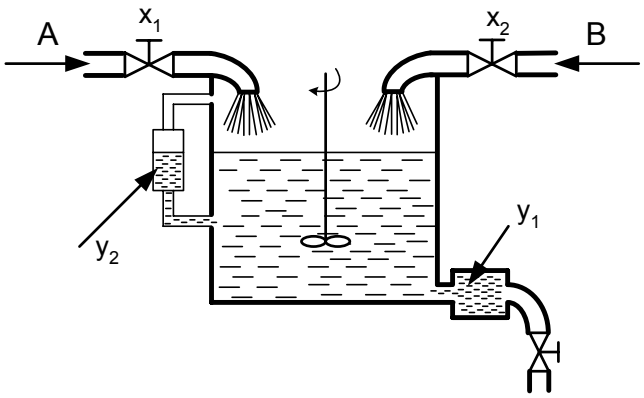
Rys. 2.7 Przykład elementu wzmacniającego

Elementy o wielu wejściach i wyjściach – w praktyce elementy posiadające jedno wejście i jedno wyjście spotykane są rzadziej niż elementy, w których występuje większa liczba wejść i wyjść. Liczba sygnałów wejściowych nie musi być równa jest liczbie sygnałów wejściowych (*ang. Multiple Input Multiple Output*, MIMO).



Rys. 2.8 Element automatyki o wielu wejściach i wyjściach

Przykładem elementu o większej liczbie wejść i wyjść może być zbiornik, do którego są doprowadzane są substancje A i B. Sygnałami wejściowymi x_1 i x_2 są przesunięcia zaworów regulujących dopływ substancji A i B. Sygnałami wyjściowymi są: y_1 pH mieszaniny oraz poziom cieczy y_2.



Rys. 2.9 Przykład elementów o kilku wejściach i wyjściach

3 Regulator PID

Podstawowe rodzaje regulatorów o działaniu ciągłym (traktowanych zwykle jako elementy liniowe) realizują funkcję typu P–proporcjonalno (*ang.* ***P****roportional*) I–całkująco (*ang.* ***I****nterval*) D–różniczkującego **(***ang.* ***D****erivative*), określone za pomocą transmitancji

$$G(s) = \frac{x(s)}{e(s)} = K_p + \frac{K_i}{s} + K_d * s \qquad [3.1]$$

po przekształceniu funkcji *G(s)*

$$G(s) = \frac{x(s)}{e(s)} = \frac{Kd * s^2 + Kp * s + Ki}{s} \qquad [3.2]$$

gdzie K_p – człon proporcjonalny, K_i –człon całkujący i K_d –człon różniczkujący.

W Matlabie strukturę równoległą[18] zapisuje się następująco:

```
PIDgain=[Kd Kp Ki];
PIDden=[1 0];
PIDreg = tf(PIDgain,PIDden);            % Trasfer function of PID Controller
```

Należy zwrócić uwagę na kolejność zapisu współczynników wzmocnienia regulatora, tj. Kd Kp Ki.

Oprócz działania PID, poszczególne wykonania regulatorów mogą realizować prostsze funkcje, będące przypadkami szczególnymi. Może występować pojedynczy człon wzmacniający Kp (człon proporcjonalny)

$$G(s) = \frac{x(s)}{e(s)} = K_p \qquad [3.3]$$

bądź proporcjonalno całkujący (PI)

$$G(s) = \frac{x(s)}{e(s)} = K_p(1 + \frac{1}{T_i * s}) \quad [3.3]$$

W automatyce spotyka się regulatory PID opisane równaniem (przekształceniem równania [3.1])

$$G(s) = \frac{x(s)}{e(s)} = K_p(1 + \frac{1}{T_i s} + T_d * s) \qquad [3.4]$$

gdzie:

[18] przykład struktury równoległej na rys. 3.4 w rozdziale 3.3

K_p – wzmocnienie proporcjonalne,

T_i – *czas zdwojenia* (stała czasowa akcji całkującej),

T_d – *czas wyprzedzenia* (stała czasowa akcji różniczkującej),

Zamiast wzmocnienia proporcjonalnego K_p podaje się często tzw. *zakres proporcjonalności,* X_p w procentach

$$X_p = \frac{1}{k_p} * 100\% \qquad [3.5]$$

Zakres proporcjonalności można rozumieć jako procentową część pełnego zakresu zmian wielkości wejściowej *e*, potrzebną do wywołania zmiany wielkości wyjściowej *x* o pełen zakres.

Czas zdwojenia T_i określa intensywność działania całkującego regulatora. Nazwa *czas zdwojenia* znajduje uzasadnienie na wykresie charakterystyki skokowej regulatora PI. W chwili $t = T_i$ składowa działania całkującego regulatora jest równa składowej działania proporcjonalnego, proporcjonalnego zatem całkowita wartość sygnału wyjściowego jest zdwojona w stosunku do początkowego przyrostu wielkości wyjściowej (w chwili $t= 0^+$), reprezentującego tylko działanie proporcjonalne.

Czas wyprzedzenia T_d określa intensywność działania różniczkującego regulatora. W dużym uproszczeniu nazwę tę można uzasadnić w następujący sposób: dzięki działaniu różniczkującemu regulator może bardzo silnie reagować już na małe zmiany odchylenia regulacji *e* przez odpowiednie oddziaływanie na obiekt regulacji.[19]

Regulator w układzie ze sprzężeniem zwrotnym jest dobrze zaprojektowany dla przypadków ogólnych gdy:

- Zapas modułu wynosi co najmniej 6÷8 dB,
- Zapas fazy wynosi co najmniej $\Pi/3$ [rad]. [20]

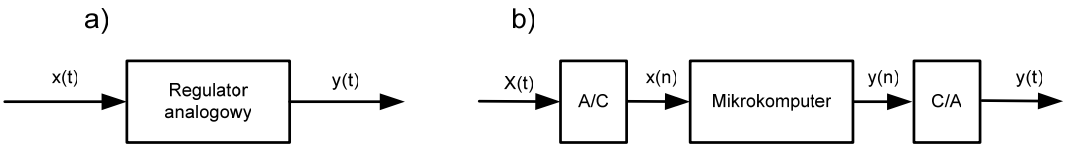
[19] Marek Żelazny, *Podstawy automatyki,* s. 144, PWN, Warszawa 1976r.

[20] Jerzy Mościcki, Zbigniew Ogonowski, *Advance Control with Matlab & Simulink,* Ellis Horwood 1995 s. 67

3.1 Regulator cyfrowy a regulator analogowy[21]

Na poniższym rysunku uwidocznione są zasadnicze różnice pomiędzy regulatorem analogowym i cyfrowym. Podstawową właściwością regulatora analogowego jest przetwarzanie w sposób ciągły sygnału wejściowego *x(t)*. Natomiast cechą charakterystyczną regulatora cyfrowego jest cykliczność jego pracy. Cykl pracy regulatora cyfrowego można podzielić na trzy etapy:

- przetworzenie pomierzonego sygnału analogowego *x(t)* wartości wyjściowej na postać cyfrową *x(n)*,
- wypracowanie, zgodnie z algorytmem, sygnału sterującego *y(n)*,
- przetworzenie sygnału cyfrowego *y(n)* na analogowy *y(t)*.



Rys. 3.1 Regulatory: a) analogowy, b) cyfrowy

Czas jaki zajmuje regulatorowi wysterowanie członu wykonawczego określony jest sumą czasów: przetwarzania A/C, obliczenia wielkości sterującej przez zapisany w pamięci program oraz przetwarzania C/A. Czasy przetwarzania są determinowane przez użyte przetworniki, natomiast o czasie wypracowania sygnału regulującego decyduje zastosowany algorytm regulatora.

Nie w każdym okresie próbkowania muszą wystąpić wszystkie przytoczone wyżej fazy działania regulatora cyfrowego. Dopuszcza się przypadki, gdzie sygnał wyjściowy może być wyznaczony z mniejszą częstotliwością niż dokonywane pomiary, a także stosuje się rozwiązania, w których okres próbkowania jest na tyle długi, iż możliwe jest sterowanie jednoczesne w kilku pętlach, bądź wykonywanie innych zadań.

[21] Beliczyński B., *Wprowadzenie do regulacji cyfrowej*, Wyd. Politechniki Warszawskiej, 1987r..

3.2 Algorytmy regulatora cyfrowego PID

3.2.1 Algorytm podstawowy regulatora cyfrowego PID

Regulator PID to najbardziej uniwersalny typ regulatora, zezwala on na regulację obiektami o różnych charakterystykach. Klasyczny regulator PID na podstawie sygnału uchybu e(t), będącego różnicą pomiędzy wartością zadaną y_z a rzeczywistą y(t) wielkości sterowanej, wypracowuje sygnał sterujący u(t) będący wynikiem trzech działań.

Zapisując konwencjonalny ciągły algorytm PID w postaci:

$$u[t]=K_p*(e[t]+1/T_i*\int e[t]dt + T_d*de[t]/dt) \quad [3.6]$$

można, używając różnic skończonych, uzyskać wprost cyfrowy algorytm regulacji PID

$$u[n]= K_p*([e[n] + T_p/T_i*\sum_{i=1} e[i-1] + T_d/T_p*(e[n]-e[n-1]) \quad [3.7]$$

gdzie:

K_p- współczynnik wzmocnienia części proporcjonalnej regulatora,

T_i - czas zdwajania

T_d - czas wyprzedzania

T_p - czas próbkowania;

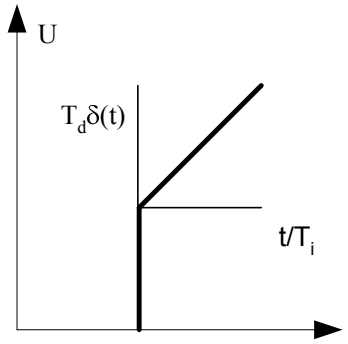
e[n] - uchyb w n-tym okresie próbkowania.

Człon proporcjonalny regulatora PID powoduje wzmocnienie uchybu i zapewnia sukcesywne jego zmniejszanie. Działanie P prowadzi do niestabilności układu regulacji. Wzrost wzmocnienia K_p powoduje zmniejszanie zapasu stabilności, poszerzenie pasma roboczego i współczynnika statyzmu. Przy skokowej zmianie sygnału regulującego występuje skokowa, proporcjonalna zmiana sygnału wyjściowego, a występujący błąd regulacji nie jest całkowicie likwidowany (tzw. statyczny uchyb regulacji).

Człon całkujący w regulatorze PID powoduje korekcję w zakresie małych częstotliwości, wprowadzając efekt astatyzmu. Człon całkujący dla większych częstotliwości zmniejsza znacznie wzmocnienie, ogranicza pasmo robocze jak również wprowadza przesunięcie fazy, równe - $\pi/2$, prowadzące do pogorszenia stabilności. Człon całkujący reaguje wolniej na zmiany sygnału sterowanego. Wielkość regulowana oscylując wokół swojej wartości średniej

sprowadza uchyb regulacji do zera. Połączone działanie PI wykorzystując zalety całkowania, eliminuje jego wady, gdyż nie wprowadza przesunięcia fazy i nie ogranicza pasma.

Człon różniczkujący stosuje się w celu przyśpieszenia przebiegów zachodzących w układzie regulacji. Sygnał wyjściowy z członu różniczkującego jest proporcjonalny do prędkości zmian sygnału wejściowego, nie zależąc od jego amplitudy. W połączeniu z elementami całkującym i proporcjonalnym, działanie członu różniczkującego wprowadza do sygnału wyjściowego z regulatora składnik zależny od prędkości zmian uchybu regulacji. Przy wzroście odchyłki regulacji, regulator wytwarza sygnał przeciwdziałający jej wzrostowi. W sumie połączenie działań P i D zapewnia zwiększenie zapasu stabilności, wzmocnienia i rozszerzenie pasma roboczego.



Rys. 3.2 Charakterystyka skokowa idealnego, ciągłego regulatora

3.2.2 Modyfikacje algorytmu regulatora cyfrowego PID

Istnieją dwie zasadnicze postacie algorytmu regulatora w sterowaniu DDC (*ang. Digital Direct Control*):

- przyrostowa,
- pozycyjna.

W algorytmie pozycyjnym, sterowanie w k-tym okresie próbkowania wyznaczane jest w oparciu o sumę uchybów z próbkowań poprzedzających.

W algorytmie przyrostowym, zwanym również prędkościowym, sterowanie w k-tym okresie próbkowania jest przyrostem sygnału sterowania w stosunku do sterowania w (k-1) - ej próbce.

Podstawowa forma algorytmu pozycyjnego regulatora PID, przy założeniu stosowania małych okresów próbkowania, może być aproksymowana równaniem różnicowym [3.7]. Takie przedstawienie algorytmu regulatora cyfrowego umożliwia stosowanie (przy optymaliza-

cji nastaw regulatora cyfrowego) metod optymalizacji nastaw regulatorów analogowych. W równaniu [3.7] pochodną zastąpiono różnicą pierwszego rzędu, natomiast całkę sumą. Zastosowano całkowanie metodą prostokątów. Jest to tzw. nierekurencyjny, pozycyjny algorytm regulatora cyfrowego PID.

W układach rzeczywistych stosuje się liczne modyfikacje powyższego algorytmu, w celu osiągnięcia poprawy wskaźników jakości sterowania, takich jak np.: czas ustalania się przebiegów przejściowych w układzie, wielkość przeregulowania, suma wartości bezwzględnych uchybów.

Z równania [3.7] można w sposób prosty wyprowadzić podstawową, przyrostową postać algorytmu regulatora PID [3.9]. Zależność [3.8] przedstawia sterowanie w (k-1)-ym okresie próbkowania w algorytmie pozycyjnym.

$$u[k-1]= K_p*[e[k-1] + T_p/T_i*\sum_{i=1}^{k-1} e[i-1] + T_d/T_p*(e[k-1]-e[k-2]) \qquad [3.8]$$

$$u[k]-u[k-1]=K_p*(1+T_d/T_p)*e[k] - K_p*(1+2T_d/T_p-T_p/T_i)*e[k-1] + K_p*T_d/T_p \qquad [3.9]$$

W powyżej przytoczonych algorytmach całkowanie przybliżano metodą prostokątów. Przy zastosowaniu całkowania metodą trapezów algorytm pozycyjny regulatora cyfrowego PID przyjmie postać:

$$u[k]= K_p*[e[k] + T_p/T_i*((e[0] + e[k])/2 + \sum_{i=1}^{k-1} e[i] + T_d/T_p(e[k]-e[k-1]) \qquad [3.10]$$

natomiast odpowiadający mu algorytm przyrostowy w postaci rekurencyjnej ma postać:

$$u[k]-u[k-1]=K_p*(1+T_d/T_p+T_d/2T_i)*e[k]-K_p*(1+2T_d/T_p-T_p/2T_i)*e[k-1]+K_p*T_d/T_p \qquad [3.11]$$

Aproksymacja metodą trapezów, w przeciwieństwie do aproksymacji prostokątnej, dla większych częstotliwości charakteryzuje się mniejszą wartością modułu całki w stosunku do modułu całki członu idealnego całkującego. Trapezowa metoda całkowania dyskretnego jest metodą dokładniejszą. Charakterystyka fazowa integratora trapezowego jest zbliżona do charakterystyki fazowej członu idealnego całkującego.

Inną modyfikacją algorytmu regulatora cyfrowego PID jest takie przekształcenie algorytmu, które prowadzi do eliminacji wpływu szybkiej zmiany wartości zadanej na wartość regulowaną. W takim przypadku w działaniu różniczkującym regulatora uwzględnia się tylko wartości zmiennej regulowanej zastępując pochodną uchybu *de(t)/dt* pochodną zmiennej regulowanej - *dy(t)/dt*. Wówczas algorytm prędkościowy sterowania ma postać:

$$u[k]-u[k-1] = K_p*\{e[k]-e[k-1] + T_p/T_i* e[k-1] + T_d/T_p*(-y[k] + 2y[k-1] - y[k-2])\}. \quad [3.12]$$

Natomiast podstawiając za wartość uchybu różnicę wartości zadanej i wartości regulowanej $e[k] = y_z[k] - y[k]$ otrzymuje się następującą postać algorytmu:

$$u[k]-u[k-1] = K_p*\{-y[k] + y[k-1] + T_p/T_i *e[k-1] + T_d/T_p*(-y[k] + 2y[k-1] - y[k-2])\}. \quad [3.13]$$

Z ostatniej modyfikacji wynika, iż nie można zrezygnować z działania całkującego, ponieważ tylko w członie całkującym występuje wartość zadana $y_z[k-1]$ i usunięcie członu całkującego spowodowałoby dryft wartości regulowanej.

Kolejne modyfikacje algorytmu cyfrowego regulatora PID polegają na różnym przedstawieniu procesu różniczkowania. Jeśli w mierzonym sygnale zmiennej regulowanej występują zakłócenia to obliczona wartość pochodnej przy pomocy różnicy pierwszego rzędu jest obarczona dużym błędem. Chcąc tego uniknąć należy np. wyznaczyć pochodną poprzez czteropunktową różnicę centralną.

Algorytm pozycyjny z zastosowaną czteropunktową różnicą centralną i całkowaniem metodą prostokątów przedstawia równanie:

$$u[k]= K_p*\{e[k] + T_p/T_i*\Sigma_{i=1} e[i-1] + T_d/6T_p*(e[k]-e[k-3]+3e[k-1]-3e[k-2])\}. \quad [3.14]$$

Podobne możliwości wyznaczenia pochodnej funkcji przebiegu daje wykorzystanie wielomianów interpolacyjnych Newtona, Gaussa, Lagrange'a. Przykładowo stosując wielomian interpolacyjny Lagrange'a, czteropunktowa pochodna przybliżona jest zależnością:

$$dy[k]/T_p= 1/6T_p(11y[k] - 18y[k-1] + 9y[k-2] - 2y[k-3]). \quad [3.15]$$

Inna modyfikacja algorytmu regulatora cyfrowego PID może polegać na porównywaniu znaków składowej proporcjonalnej oraz składowej proporcjonalnej całkującej regulatora. Gdy zmienna regulowana zbliża się do wartości zadanej bardzo często znaki wymienionych składowych są różne, natomiast przy oddalaniu się znaki obydwu składowych są takie same. Właściwość ta wykorzystywana jest przy wartościach zmiennych regulowanych bliskich war-

tości zadanej, ponieważ eliminowane jest wówczas przeregulowanie, generowane przez człon całkujący. Przy dużych wartościach uchybu własność powyższa wydłuża czas regulacji. Chcąc temu zapobiec, wprowadza się wokół wartości zadanej strefę o szerokości około 5% pełnego zakresu zmian wielkości regulowanej y_{max}, poza którą uwzględnia się tylko te składowe proporcjonalne, które mają ten sam znak co składowa całkująca.

W praktyce można również spotkać modyfikację wykorzystującą tzw. regulator strefowy, w którym wartości współczynników nastaw są zależne od wartości uchybu regulacji. Przykładowo, dla większych wartości uchybu regulacji od $0.05*y_{max}$, wprowadza się większy współczynnik wzmocnienia, natomiast dla wartości uchybu mniejszych od ustalonego progu wartości, wartość współczynnika wzmocnienia zmniejsza się.

Z algorytmu cyfrowego regulatora PID można w sposób prosty otrzymać, w zależności od potrzeb, algorytm cyfrowy regulatorów P, PI, PD. Należy jedynie wyzerować odpowiednie składowe. Regulator P uzyskamy usuwając części całkującą i różniczkującą.

Algorytm regulatora P w postaci pozycyjnej przedstawia równanie [3.16].

$$u[k] = K_p * e[k]. \qquad [3.16]$$

Z algorytmu, tak jak to wspomniano wcześniej, nie można usuwać samej części całkującej, gdyż jest ona jedyną częścią zawierającą wartość zadaną, redukującą się w części różniczkującej i proporcjonalnej.

Eliminując część różniczkującą z równania [3.7] otrzymamy pozycyjną postać regulatora cyfrowego PI [3.17].

$$u[k] = Kp*(e[k] + Tp/Ti*\sum_{i=1}^{k} e[i]). \qquad [3.17]$$

Algorytm pozycyjny cyfrowego regulatora PD można opisać równaniem

$$u[k] = Kp*(e[k] + Td/Tp * (e[k] - e[k-1]). \quad [3.18]$$

Podsumowując powyżej przedstawione postacie pozycyjnych i przyrostowych algorytmów regulatorów cyfrowych należy stwierdzić że:

- Wybór przy sterowania DDC jednej z form algorytmu zdeterminowany jest przede wszystkim strukturą sterowanego obiektu.
- Algorytm przyrostowy należy stosować w przypadku, gdy na wejściu sterowanego obiektu znajduje się człon całkujący. W przypadku takim realizacja całkowania poza regulatorem zwalnia pamięć komputera od uczestnictwa w wykonywania procesu całkowania.

- Sterowanie przy zastosowaniu algorytmu przyrostowego zapobiega przeregulowaniu całkowemu, które charakteryzuje się narastaniem wartości sumy uchybów przy nasyconym członie wykonawczym.
- Zaletą stosowania algorytmu pozycyjnego jest fakt, iż wartość sygnału sterującego przechowywana w pamięci komputera jest niezbędna przy nieliniowych modyfikacjach algorytmu oraz przy dodatkowych sprzężeniach korekcyjnych.

3.3 Struktury regulatora cyfrowego PID

Istnieje wiele sposobów realizacji algorytmu regulatora PID o równaniu podanym wzorem [3.7]. Właściwości PID można uzyskać za pomocą rozmaitego połączenia członów P, I, i D. Wybór określonej struktury regulatora jest zdeterminowany trzema głównymi czynnikami:

- zakresem nastaw dającym się uzyskać przy danej strukturze,
- ograniczeniem nastaw polegającym na tym, że w wybranej strukturze nie może być możliwe jednoczesne nastawianie dowolnych wartości nastaw,
- zależnością nastaw, zwaną również interakcją, polegająca na wzajemnej zależności między nastawami.

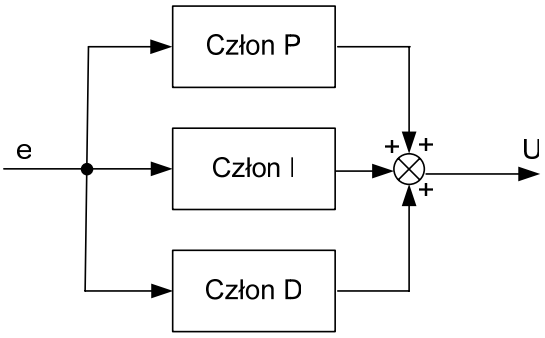
Wyróżnia się trzy grupy wśród stosowanych powszechnie struktur[22]:

- Grupę pierwszą stanowią układy dające się przedstawić połączeniem równoległym trzech członów składowych regulatora. W strukturze równoległej o trzech gałęziach, człon o nastawianym współczynniku Kp jest umieszczony przed rozgałęzieniem lub za rozgałęzieniem, w celu uzyskania całkowitą niezależności nastaw. Struktury tej grupy nie mają ograniczenia nastaw.
- Drugą grupę struktur stanowią rozwiązania, w których jest tylko jeden wzmacniacz o możliwie dużym współczynniku wzmocnienia, objęty dynamicznym sprzężeniem zwrotnym.
- Trzecia grupa struktur charakteryzuje się tworzeniem kombinacji z członów PI oraz PD, możliwe jest ich połączenie szeregowe lub równoległe. W połączeniu szeregowym pewną rolę odgrywa kolejność członów (jeżeli rozpatrywać wpływy

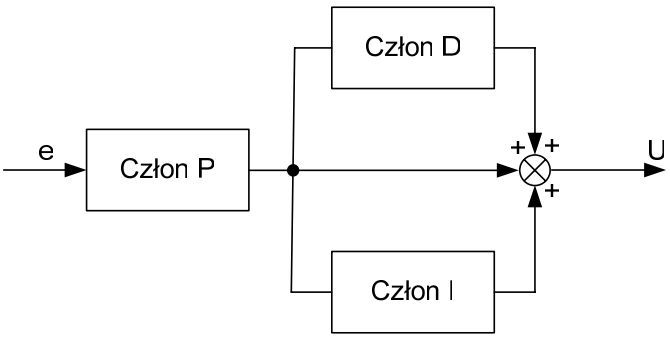
[22] W. Findeisen, *Technika regulacji automatycznej*, PWN Warszawa, 1978r.

ograniczeń nastaw oraz problem przeregulowania całkowego). W strukturze równoległej nie występują ograniczenia nastawy.

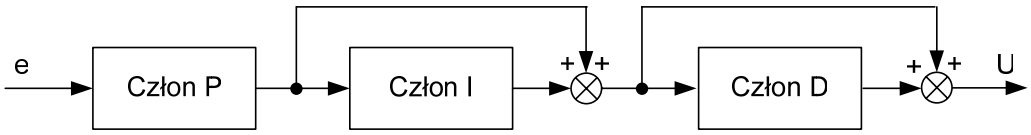
W programie wykorzystano idealną równoległą strukturę regulatora PID. Występują również struktury: szeregowo-równoległe (nazywaną inaczej równoległą o trzech gałęziach), oraz interakcyjną.



Rys. 3.3 Idealna-równoległa struktura regulatora PID



Rys. 3.4 Szeregowo-równoległa struktura regulatora PID.



Rys. 3.5 Interakcyjna struktura regulatora PID

3.4 Czy regulator PID ma przyszłość?[23]

Łatwo przewidzieć, że regulator PID będzie stosowany w przyszłości. Sprzężenie zwrotne ma rewolucyjny wpływ w praktycznie wszystkich dziedzinach. Regulator PI(D) jest prawdopodobnie najbardziej podstawowa formą sprzężenia zwrotnego. Jest bardzo skuteczny i może być stosowany w szerokim zakresie problemów. Możliwość automatycznego strojenia regulatorów PID bardzo uprościła sterowanie.

Podstawowa wiedza na temat PID jest dostępna od wielu lat. Zainteresowanie regulatorami PID znacznie wzrosło w ciągu ostatnich 10 lat. Proces ten można zaobserwować na poniższym wykresie. Wiele artykułów na ten temat zostało opublikowanych.[24]

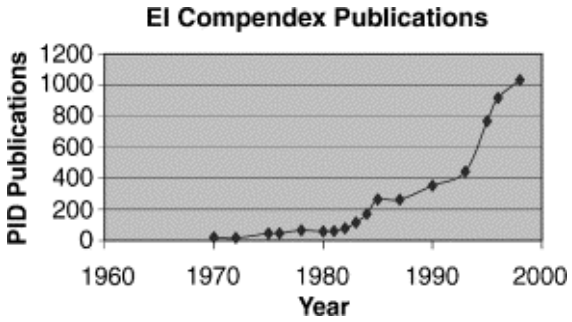
Alternatywą dla regulatora PID są:

RST Discrete–time linear MISI controllers

SFO State feedback and observers

MPC Model predictive control[25]

Regulatory rozmyte (*ang. Fuzzy control*) są również często proponowane jako alternatywa dla regulatorów PID. Większość regulatorów rozmytych używanych w przemyśle posiada tą samą strukturę co regulator PI lub PID. Parametryzacja używa podstawowych reguł oraz rozmytych reguł funkcji przynależności.



Rys. 3.6 Ewolucja historyczna artykułów na temat regulatorów PID przez ostatnie 30 lat

Na powyższym wykresie można zauważyć silny wzrost zainteresowania regulatorami PI(D) Pojawiło się wiele publikacji z tej dziedziny. Regulator PI(D) zdobywa swoją popular-

[23] Control Engineering Practice Volume 9, Issue 11 listopad 2001 strony 1163-1175

[24] Astrom K.J., Hagglund, *The Future of PID control*, Control Engineering Practice 9 (2001) pages 1163-1175

[25] W celu uniknięcia niejasności pozostawiono angielską nomenklaturę.

ność dzięki swojej prostocie oraz niezawodności.[26] Listopadowy numer *Control Engineering Practice z* 2001 roku, (Volume 9, Issue 11, Pages 1159–1266) został całkowicie poświęcony regulatorom PID.

[26] Control Engineering Practice Volume 9, Issue 11 November 2001, *PID control*, p. 1160

4 Metody strojenia regulatora PID

Pomimo wielu technik strojenia regulatorów PI oraz PID opisanych w literaturze[27] oraz rozwoju nowych metodologii[28], zainteresowanie tym zagadnieniem nie słabnie. Wynika to niewątpliwie z powodu szerokiego zastosowania regulatorów w przemyśle, jak również z faktu, iż część z nich jest źle strojona.[29] Jednakże, większość obecnych metod strojenia parametrów regulatorów PI oraz PID nie znajduje optymalnych nastaw.[30] Rzeczywiste obiekty wysokiego rzędu, występujące w przemyśle są opisywane za pomocą transformaty pierwszego lub drugiego rzędu z opóźnieniem transportowym lub bez niego. Ponadto, jeśli złożoność procesów wzrasta, np. niestabilny obiekt pierwszego rzędu lub obiekty z opóźnieniem transportowym, ilość odpowiednich metod strojenia maleje lub w wielu wypadkach zanika. Metody strojenia używane w tego typu procesach nie opierają się na wydajności oraz niezawodnych kryteriach, a zatem otrzymane parametry nie zawsze są optymalne. Natomiast w przypadku zastosowania algorytmów genetycznych do regulatora PID, jako funkcję przystosowania przyjmuje się kryteria błędów, czyli optymalizuje się wydajność regulatora poprzez minimalizację funkcji kosztów.

4.1 Algorytmy genetyczne

Istnieje wiele metod na poszukiwanie układów optymalnych i jedną z nich są Algorytmy Genetyczne, AG. Układem optymalnym nazywamy układ najlepszy w sensie wybranego kryterium oceny. Pod pojęciem sterowania optymalnego rozumie się takie sterowanie dopuszczalne, które przy danych równaniach obiektu i ograniczeniach minimalizuje wskaźnik jakości. Należy postawić pytanie: dlaczego padł wybór na algorytmy genetyczny, jeśli posia-

---

[27] zagadnienie rozwinięte w *Control Engineering Practice*, Volume 9, Issue 11, Pages 1159-1266 November 2001

[28] zob. na przykład, Jialiang Lu, Guanrong Chen andHao Ying*, Predictive fuzzy PID control: theory, design and simulation***,** Information Sciences, Volume 137, Issues 1-4, September 2001, Pages 157-187

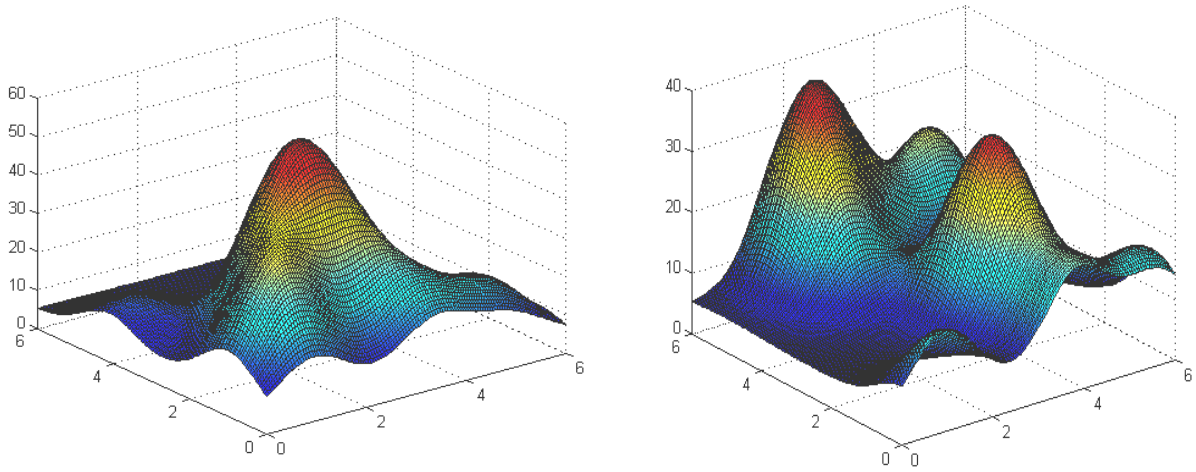
[29] Ender, D.B., *Process control performance: not as good as you think,* Control Engineering, September 1993r. pages 180-190.

[30] O'Mahony Tom, C.J. Downing, Fatla Klaudiusz, *Genetic Algorithms for PID Parameter Optimisation: Minimising Error Criteria*

damy inne sprawdzone metody? Kiedy zaczniemy klasyfikować problemy, to możemy podzielić je na dwie grupy:

- problemy rozwiązywalne,
- problemy nierozwiązywalne.

W rzeczywistości to, co jest rozwiązywalne, może nie być rozwiązane w rozsądnym czasie (np. czas rozwiązania może być bliski nieskończoności). Oczywiście, że złożoność obliczeń ma też dobre strony, jedną z nich jest kryptografia, gdzie wykorzystuje się to zjawisko dość powszechnie. Obecnie wyróżnia się trzy rodzaje metod optymalizacyjnych: analityczne, enumeratywne (przeglądowe) i losowe.



Rys. 4.1 Przykłady optymalizowanych funkcji

Metody analityczne dzieli się na bezpośrednie i pośrednie. W metodach pośrednich poszukujemy ekstremów lokalnych rozwiązując układ równań (zazwyczaj nieliniowych) otrzymanych przez przyrównanie gradientu do zera. Jeśli mamy funkcję gładką określoną na otwartym zbiorze, to szukanie ekstremów jest zadaniem łatwym stosując metodę wspinaczki (wspinaj się wzdłuż najbardziej stromej z możliwych dróg). Jeśli natomiast funkcja z rys 4.1 umieszczonego po lewej stronie, jest wycinkiem z rysunku umieszczonego po prawej stronie to, jeśli rozpocznie się wspinaczkę z sąsiedztwie niższego wierzchołka, przeoczy się wierzchołek wyższy. Co gorsza, nie będzie możliwości przejścia z wierzchołka niższego na wyższy bez powtórzenia całego procesu. Stosowanie tych metod jest uzależnione od istnienia pochodnych lub numerycznej ich aproksymacji. Wiele spotykanych w rzeczywistości przestrzeni parametrów nie wykazuje respektu dla istnienia pochodnej i są pełne nieciągłości.

Metody przeglądowe opierają się na następującej idei: mając skończoną przestrzeń poszukiwań, algorytm oblicza funkcje celu i przegląda wszystkie punkty po kolei. Jest to zbli-

żone do ludzkiego sposobu rozwiązywania problemu. Sposób ten jest dość prosty, ale nieefektywny dla dużych przestrzeni.

Metody losowe zaczęły się cieszyć coraz większą popularnością, kiedy zdano sobie sprawę z ograniczeń poprzednich metod. Jednak błądzenie przypadkowe należy zdyskwalifikować ze względu na efektywność.

Algorytmy genetyczne są stochastyczną metodą poszukiwania globalnego minimum opartą na mechanizmie selekcji i genetyki. Są to metody iteracyjne, szeroko używane w optymalizacji problemów w wielu dziedzinach nauki i technologii. W przeciwieństwie do innych metod, w każdej iteracji (generacji) rozważany jest nie tylko jeden punkt płaszczyzny, ale także zbiór możliwych rozwiązań lub populacja osobników. Te osobniki są porównywane ze sobą, zgodnie z jakością rozwiązania przeprowadzoną w każdej generacji.[31] Algorytmy genetyczne różnią się od metod tradycyjnych tym że:

- nie przetwarzają bezpośrednio parametrów zadania, lecz ich zakodowaną postać,
- prowadzą poszukiwania wychodząc nie z pojedynczego punktu, lecz pewnej ich populacji,
- korzystają tylko z funkcji celu, nie zaś jej pochodnych lub innych pomocniczych informacji,
- stosują probabilistyczne, a nie deterministyczne reguły wyboru.

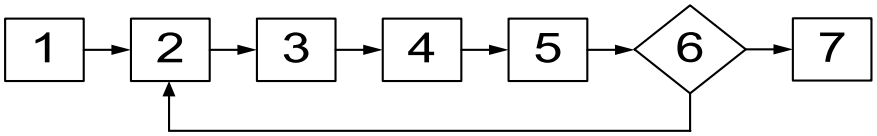
4.1.1 Charakterystyka algorytmu genetycznego

W roku 1975 John Holland z University of Michigan sformułował pierwszy algorytm genetyczny, który był oparty na mechanizmach rządzących ewolucją. Algorytm genetyczny[32] jest procesem stochastycznym przeszukującym przestrzeń rozwiązań. Jego konstrukcja sprawia, iż jest on bardzo użytecznym narzędziem o szerokich zastosowaniach. Proces poszukiwawczy zaczyna się od losowo wybranego elementu, a w następnej generacji tworzone są lepsze rozwiązania. Nowe rozwiązania są rezultatem kilku podstawowych operacji, które naśladują procesy zachodzące w naturze. W tej dziedzinie i w genetyce spotykamy takie same pojęcia. Każdy element zbioru rozwiązań jest nazwany ***chromosomem***, *a* część chromosomu ***genem***.

[31] *Mathematics and Computers in Simulation* Volume 51, Issues 3-4 January 2000 Pages 287-300

[32] szerzej w czasopiśmie *Artificial Intelligence in Engineering* wydawanym przez Elsevier Science Ltd.

Wszystkie chromosomy zwane są ***populacją*** *a* operacje używane to tworzenia następnej generacji osobników to ***krzyżowanie*** (*ang. crossover*) i ***mutacja***. Jeżeli do rozwiązania problemu chcemy wykorzystać możliwości dane przez algorytmy genetyczne, to ten problem musimy opisać w odpowiedni sposób. Problem kodujemy jako ciąg binarny, który tworzą chromosomy. Sposób zakodowania informacji ma bardzo duży wpływ na efektywność algorytmu genetycznego. Reguły selekcji naturalnej są tu zastąpione przez funkcję przystosowania. W oparciu o tę funkcję algorytm rozpoznaje, które z osobników lepiej pasują jako rozwiązanie problemu i mają szansę dać jeszcze lepsze rozwiązanie w następnej populacji. Następnym krokiem jest reprodukcja, podczas której wybierane są osobniki o największej wartości funkcji przystosowania. Potem następuje krzyżowanie, którego miejsce jest określane losowo. W tym samym czasie zachodzi zjawisko zwane mutacją. Wynikiem mutacji jest sytuacja, kiedy „potomek" jest skrajnie różny od „rodziców", więc z tego względu prawdopodobieństwo mutacji powinno być małe. W tym miejscu następuje zamknięcie pętli ewolucyjnej, co zostało zilustrowane na rysunku poniżej. Reprodukcja zaczyna się znów od miejsca gdzie są wybierane osobniki do tworzenia nowej generacji. Wybór chromosomów jest zjawiskiem przypadkowym, ale bardzo ściśle ukierunkowanym na wybieranie najlepszych osobników. Typową praktyką jest zakończenie procesu szukania po wygenerowaniu określonej liczby pokoleń lub wtedy, kiedy najlepszy z osobników spełni założone przez nas wymagania.[33] Jeśli algorytm nie znajdzie optymalnego rozwiązania, to algorytm jest uruchamiany od początku z nową populacją początkową.



Rys. 4.2 Pętla algorytmu genetycznego

1. Populacja początkowa,
2. Wybór osobników do reprodukcji,
3. Krzyżowanie wybranych osobników,
4. Mutowanie nowego pokolenia,
5. Tworzenie nowej populacji,
6. Czy spełniono wymagania?
7. Koniec poszukiwań.

[33] zob. plik genhaploid.m w załączniku A. Polecenie `options` *(14)* przypisuje maksymalną wartość generacji, natomiast polecenie `options` *(13)* przerywa działanie algorytmu po spełnieniu satysfakcjonujących nas warunków.

4.1.2 Biologiczne podstawy AG

W sztucznych systemach genetycznych łańcuch chromosomu jest analogiczny jak w systemie biologicznym. Wszystkie łańcuchy tworzą jedną strukturę, która opisuje dokładnie jedno rozwiązanie, alternatywne rozwiązania lub punkt w przestrzeni rozwiązań.[34] Łańcuchy określają cechy lub detektory, które przyjmują różne wartości. Cechy mogą być zlokalizowane na rożnych pozycjach łańcucha.

W rzeczywistym systemie jeden lub więcej chromosomów tworzą całkowity obraz budowy i działania organizmu. Całość materiału genetycznego nazywana jest ***genotypem***. Natomiast organizm opisany genotypem wraz ze swoim otoczeniem nazywany jest ***fenotypem***. Chromosomy połączone są w ***geny***, które mogą przyjmować kilka wartości danej cechy, zwanej ***allelem***. Pozycja genu, jego umiejscowienie, jest rozpatrywana oddzielnie od funkcji genu. Na przykład dla zwierząt gen odpowiedzialny za kolor oczu jest umiejscowiony na pozycji 10, natomiast wariant cechy (allela) to kolor niebieski.

4.1.3 Reprezentacja chromosomu

Osobniki są przedstawione jako łańcuchy, chromosomy[35], skonstruowane według pewnych zasad. Tak więc genotyp jest unikalnie odwzorowany jako zmienna. Najczęściej używanym systemem zapisu przy konstrukcji algorytmu genetycznego, jest system binarny. Zatem każda ze zmiennych jest kodowana jako łańcuch dwójkowy. Chromosomy są reprezentowane przez ciąg znaków binarnych o określonej długości i zamieniane na wartości rzeczywiste poprzez standardową zamianę systemów lub kod Gray'a. Używanie kodu Gray'a jest zalecane, ponieważ poszukiwanie staje się mniej zwodnicze.

[34] zob. rys 4.1

[35] zob. rys 4.5

4.1.4 Populacja

Kiedy zdecydujemy, jaki typ reprezentacji chromosomu zostanie wykorzystany, to pierwszym krokiem w AG jest stworzenie populacji początkowej. Często populację początkową tworzy się poprzez wykorzystanie generatora liczb losowych, który wybiera elementy ze zbioru o zadanych granicach, np.: dla reprezentacji binarnej *N* osobników, których chromosom na *L* bitów długości, to musi być wybranych *N x L* liczb losowych ze zbioru {0, 1}.

Istnieją trzy podstawowe rodzaje tworzenia nowej populacji: prosty, ustabilizowany i wymuszony.

4.1.4.1 Prosty AG[36]

Ten typ populacji jest często wykorzystywany do implementacji algorytmu genetycznego. Ten typ algorytmu genetycznego wykorzystuje niepokrywającą się populację. W każdym pokoleniu cała populacja jest zastępowana przez nową, żaden z osobników z poprzedniego pokolenia nie zostaje zachowany.

4.1.4.2 Ustabilizowany AG[37]

Populacja nowa zastępuje tylko część osobników z poprzedniego pokolenia, najczęściej są to osobniki najgorzej przystosowane. W ekstremalnych przypadkach tylko jeden lub dwa osobniki mogą być zastąpione, z drugiej jednak strony ten typ algorytmu można przekształcić w algorytm prosty.

4.1.4.3 Wymuszony AG[38]

Ten rodzaj algorytmu genetycznego jest bardzo podobny do powyższego z tą różnicą, że nie są zastępowane najgorzej przystosowane osobniki, ale osobnik najbardziej podobny i tylko w takim przypadku, kiedy jego przystosowanie jest mniejsze od nowego.

---

[36] *ang. Simple Genetic Algorithm*

[37] *ang. Steady state Genetic Algorithm*

[38] *ang. Struggle Genetic Algorithm*

4.1.5 Funkcja przystosowania

Funkcja przystosowania[39] (*ang. fitness function*) musi rozwiązać następujący problem: dla konkretnego chromosomu musi zwrócić konkretną wartość liczbową, która jest proporcjonalna do zdolności lub użyteczności osobnika reprezentowanego przez danych chromosom. Idealna funkcja przystosowania powinna być gładka i regularna, tak że chromosomy o lepszym przystosowaniu powinny znajdować się bliżej środka przestrzeni poszukiwań. Dla wielu problemów nie zawsze można skonstruować taką funkcję i wystarczy, aby funkcja przystosowania nie miała za dużo lokalnych maksimów lub jednego, ale bardzo odseparowanego maksimum.[40] Plik o nazwie `genpid.m`[41] ma zaimplementowaną funkcję przystosowania dla różnych kryteriów całkowych:

- Całka uchybu bezwzględnego[42] (*ang. Integral of absolute value of error*)

$$IAE = \int_0^\infty [e(t)]\ dt$$

- Całka kwadratu uchybu[43] (*ang. Integral of error squared*)

$$ISE = \int_0^\infty [e(t)]^2\, dt$$

- Całka czasu i uchybu bezwzględnego (*ang. Integral of time absolute error or Time–weighted*)

$$ITAE = \int_0^\infty t[e(t)]\ dt$$

Najprostszym kryterium jest IAE (błąd bezwzględny), które bierze pod uwagę tylko wartość bezwzględną uchybu w czasie. Błąd średniokwadratowy (ISE) jest najczęściej wykorzystywanym kryterium do optymalizacji regulatorów, jednak kładzie ono nacisk na czas narostu a nie zwraca uwagi na wielkość przeregulowania. ITAE charakteryzuje się tym, że układ z regulatorem dobrany na podstawie tego kryterium, jest układem dość szybkim, o niewielkich przeregulowaniach i jest dość odporny na zakłócenia.

---

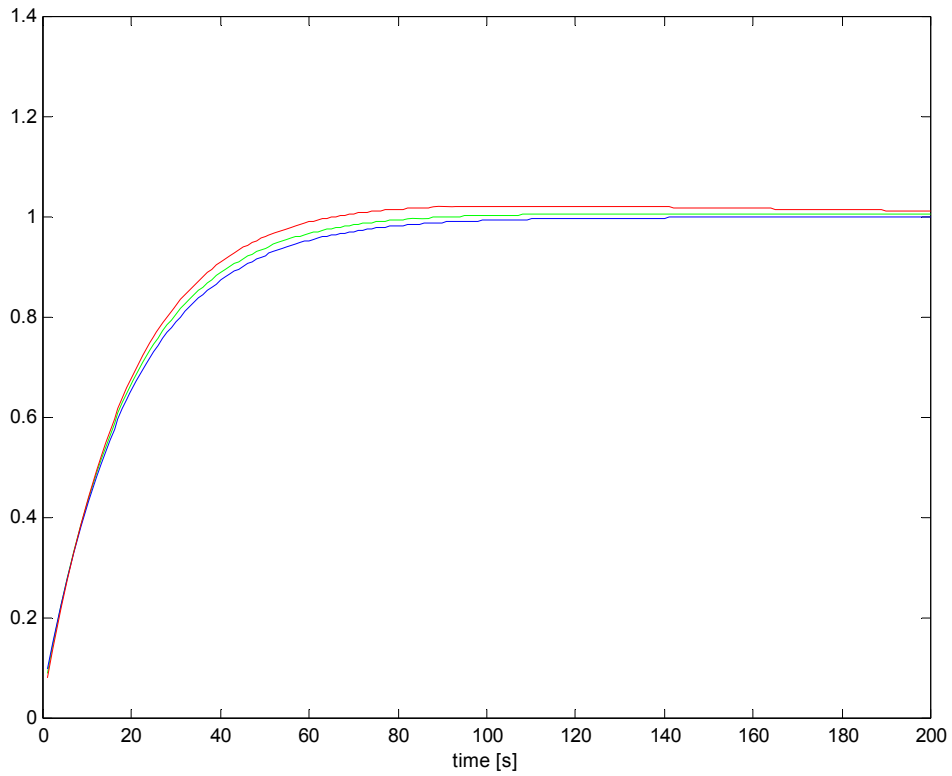
[39] W literaturze można również spotkać inne nazwy takie jak: funkcja kosztów (*ang. cost function*), funkcja celu, funkcja minimalizowana bądź optymalizowana.

[40] zob. rys. 4.1

[41] zob. załącznik A

[42] spotykana jest również forma „błąd bezwzględny”

[43] spotykana jest również forma „błąd średniokwadratowy”



Rys. 4.3 Odpowiedź układu na skok jednostkowy dla różnych funkcji przystosowania IAE (czerwony) ISE (zielony) ITAE (niebieski)

Na pytanie, które kryterium jest najlepsze, nie można jednoznacznie odpowiedzieć. Zależy to od struktury oraz zdefiniowanego problemu. Środowisko Matlab operuje na macierzach, a więc na elementach skończonych.[44] Sposób przedstawienia kryteriów całkowych w Simulinku zamieszczono w załączniku C2. Natomiast implementację funkcji przystosowania dla różnych kryteriów błędów ilustruje poniższy kod:

```
if ErrorEstim==1
  error = abs(error);                %wartość bezwzględna
   error = sum(error);               %całka
   errorIAE=error;
   disp('IAE')
   legend(num2str(errorIAE),'IAE');
elseif ErrorEstim==2
   error = error.^2;
   error = abs(error);
   error = sum(error);
   errorISE=error;
   disp('ISE')
   legend(num2str(errorISE),'ISE');
```

[44] Jerzy Mościcki, Zbigniew Ogonowski, *Advance Control with Matlab & Simulink,* Ellis Horwood 1995r., s. 23

```
elseif ErrorEstim==3
   error = error.*t;
   error = abs(error);
   error = sum(error);
   errorITAE = error;
   disp('ITAE')
   legend(num2str(errorITAE),'ITAE')
end
fitness = error;
```

4.1.6 Operacje na AG

4.1.6.1 Selekcja

Celem selekcji w algorytmie genetycznym jest danie szansy tym osobnikom, które są lepiej przystosowane. Istnieje wiele sposobów realizowania selekcji, ale najbardziej popularną techniką jest selekcja za pomocą odpowiednio wykalibrowanej ruletki. Ruletka składa się z trzech podstawowych kroków:

1. sumowanie przystosowania wszystkich osobników w populacji i nazwanie wyniku całkowitym przystosowaniem,

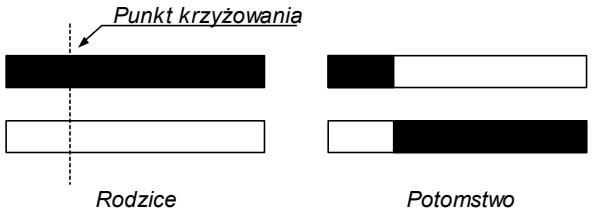
2. wygenerowanie *n* losowych liczb z przedziału {0, całkowite przystosowanie},

3. wybranie pierwszego osobnika populacji, którego przystosowanie, dodane do przystosowania poprzednich osobników, jest większe lub równe *n*.

Efektem działania ruletki jest zwrócenie losowo wybranych rodziców. Używając tego algorytmu wyboru, szansa każdego osobnika na wybór jest wprost proporcjonalna do jego wartości przystosowania. Oczywiście istnieje szansa na wybranie najgorszego osobnika w populacji. Inną bardzo popularną metodą selekcji jest uniwersalne stochastyczne próbkowanie, które opiera się na prostym algorytmie próbkującym z minimalnym zasięgem i zerowym odchyleniem.

4.1.6.2 Rekombinacja

Podstawową operacją przy tworzeniu nowych chromosomów jest krzyżowanie (*ang. crossover*). Podobnie jak w przyrodzie efektem krzyżowania są nowe osobniki, które mają fragmenty materiału genetycznego swoich rodziców. Najprostszą formą krzyżowania jest krzyżowanie proste, którego zasadę zilustrowano na poniższym rysunku



Rys. 4.4 Krzyżowanie proste

4.1.6.3 Mutacja

Mutacja powoduje zmianę reprezentacji chromosomu w oparciu o rachunek prawdopodobieństwa. Jeżeli chromosom jest reprezentowany przez ciąg znaków binarnych, to mutacja zmienia stan losowo wybranego bitu ($0 \Rightarrow 1$ lub $1 \Rightarrow 0$), co pokazano na poniższym rysunku. W procesach zachodzących w naturze mutacja jest procesem przypadkowym i zachodzi z bardzo małym prawdopodobieństwem, typowo w zakresie od 0.001 do 0.01. Mutowany jest ciąg w nowym pokoleniu i zachodzi możliwość zmutowania więcej niż jednego bitu w danym chromosomie.

Miejsce mutacji ↘
Ciąg wejsciowy: 01010110
Ciąg zmutowany:01110110

Rys. 4.5 Mutacja binarna

4.1.6.4 Nowa populacja

Selekcja i reprodukcja osobników ze starej populacji tworzy nową populacje, której przystosowanie może być zdeterminowane. Jeśli jest otrzymywane mniej osobników liczebności rekombinacji niż liczebności populacji wejściowej, wtedy różnica między nowa a starą populacją jest nazywana luką generacyjną. Aby utrzymać liczebność populacji wejściowej, nowe osobniki są wsadzane w starą populację. Podobna sytuacja zachodzi, kiedy jest tworzone więcej osobników niż wymaga tego rozmiar populacji, wtedy należy wybrać, które z osobników pozostaną. Kiedy wybieramy, które z osobników starej populacji powinny być podmienione, najlepszym wyborem jest zamiana osobników o najmniejszym przystosowaniu. Schemat wymiany osobników powinien także wybierać i podmieniać najstarsze osobniki w danej populacji.

4.1.7 Algorytmy genetyczne w Matlabie

Do zasymulowania pętli algorytmu genetycznego zastosowano *The Genetic Algorithms Toolbox for MATLAB* [45], który został napisany przez Andrew Potvina. Pakiet ten jest zbiorem gotowych procedur umożliwiających implementację algorytmów genetycznych jako macierzy, które są podstawą działania programu MATLAB. Toolbox ten składa się z następujących funkcji[46]:

- **B10TO2** zamienia ciąg z dziesiątkowego na dwójkowy,
- **DECODE** konwertuje z cyfrowego formatu na postać zmiennej,
- **ENCODE** konwertuje zmienną na format cyfrowy. Ten skrypt demonstruje użycie prostego algorytmu,
- **GENETIC** maksymalizuje funkcję kosztów używając prostego algorytmu genetycznego,
- **GENPLOT** graficzne przedstawienie drogi osiągnięcia celu przez AG,
- **MATE** przypadkowo przestawia osobniki OLD_GEN,
- **MUTATE** zmienia gen OLD_GEN z prawdopodobieństwem Pm,
- **REPRODUC** wybiera osobniki proporcjonalnie do ich funkcji przystosowania,
- **TESTGEN** testuje AG dla przykładowej funkcji $f(x)=x^{10}$,
- **XOVER** tworzy NEW_GEN z OLD_GEN używając krzyżowania.

W celu uzyskania informacji na temat składni poszczególnych funkcji należy wpisać w wierszu poleceń Matlaba *help {nazwa funkcji}* np. *help genetic.*

MATLAB jest interakcyjnym środowiskiem do wykonywania obliczeń naukowych i inżynierskich. Umożliwia on testowanie algorytmów, modelowanie i symulację, analizę i wizualizację danych, sygnałów oraz wyników obliczeń. Środowisko języka MATLAB jest otwarte, co daje możliwość wykorzystania istniejącego i budowy własnego wyspecjalizowanego oprogramowania. Simulink, który jest częścią składową pakietu MATLAB, pozwala na modelowanie i symulację ciągłych oraz dynamicznych systemów dyskretnych takich jak regulator PID.

---

[45] dostępne na serwerze producenta MATLABa
ftp://ftp.mathworks.com/pub/tech-support/solutions/s1248/genetic/

[46] nazwa funkcji jest tożsama z nazwą pliku

4.2 Komputerowe wspomaganie projektowania układów nieliniowych (*ang. Nonlinear Control Design Blockset, NCD*)

Biblioteka NCD (*ang. Nonlinear Control Design Blockset, NCD*) pakietu Matlab/Simulink umożliwia dobór optymalnych parametrów regulatora lub innych poszukiwanych optymalizowanych parametrów przy wykorzystaniu wbudowanych[47] algorytmów optymalizacji. W pracy do zagadnienia optymalizacji wykorzystano następujące oprogramowanie: pakiet *Matlab ver. 5.3, Simulink ver. 3.0.1 Optimization Toolbox ver. 2.0* oraz *NCD Blockset ver. 1.1.3*.

Metodyka syntezy regulatora PID przy wykorzystaniu NCD Blockset przedstawiona zostanie na przykładach podanych w rozdziale 5.3. Korekty układów można dokonywać regulatorami konwencjonalnymi. Wśród tradycyjnych metod, kryteriów i wskaźników jakości układów regulacji, umożliwiających dokonanie syntezy parametrycznej regulatorów, znajdują się:

- metoda Zieglera – Nicholsa[48],
- kryterium stabilności aperiodycznej,
- kryterium optymalnego modułu,
- parametry odpowiedzi skokowej układu,
- całkowe wskaźniki jakości[49],
- metoda inwersji dynamicznej.

Metody te są znane i dobrze opisane w literaturze, żadna z nich nie jest jednak tak prosta w użyciu, jak metoda syntezy regulatora w środowisku Matlab/Simulink przy użyciu NCD Blockset. Użycie NCD Blockset pozwala ponadto na równie dobrą lub lepszą jakościowo syntezę regulatora w porównaniu do wyszczególnionych powyżej metod tradycyjnych.

W celu zastosowania NCD Blockset należy połączyć blok NCD do modelu utworzonego w Simulinku. Blok NCD automatycznie konwertuje charakterystykę z dziedziny czasu na problem optymalizacyjny i rozwiązuje go używając metodę *state–of–art optimization*[50]. Problem optymalizacyjny jest sformułowany przez NCD i przywołuje symulację. Toolbox NCD korzysta z funkcji z *Optimization toolbox*, za pomocą którego są prowadzone obliczenia

[47] biblioteka optimization toolbox

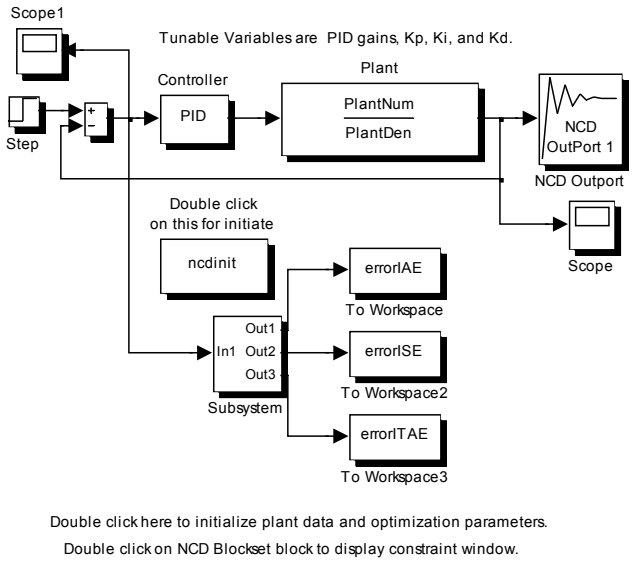
[48] opisana w rozdziale 4.3

[49] opisane w rozdziale 4.1.5

[50] Metoda ta została opisana w *Matlab Optimization Toolbox User's Guide.*

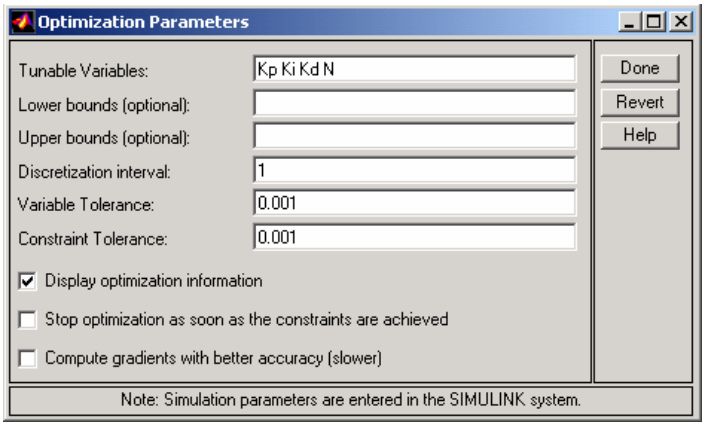
w środowisku Matlab. NCD toolbox działa tylko i wyłącznie w połączeniu z *Optimization toolbox*. Metoda optymalizacyjna jest metodą iteracyjną.

Do utworzonego w Simulinku modelu dołączono blok regulatora PID oraz blok NCD Outport. Tak zmodyfikowany układ przedstawiono na poniższym rysunku.



Rys. 4.6 Model rozpatrywanego układu dynamicznego zmodyfikowany na potrzeby syntezy regulatora PID przy wykorzystaniu NCD Blockset

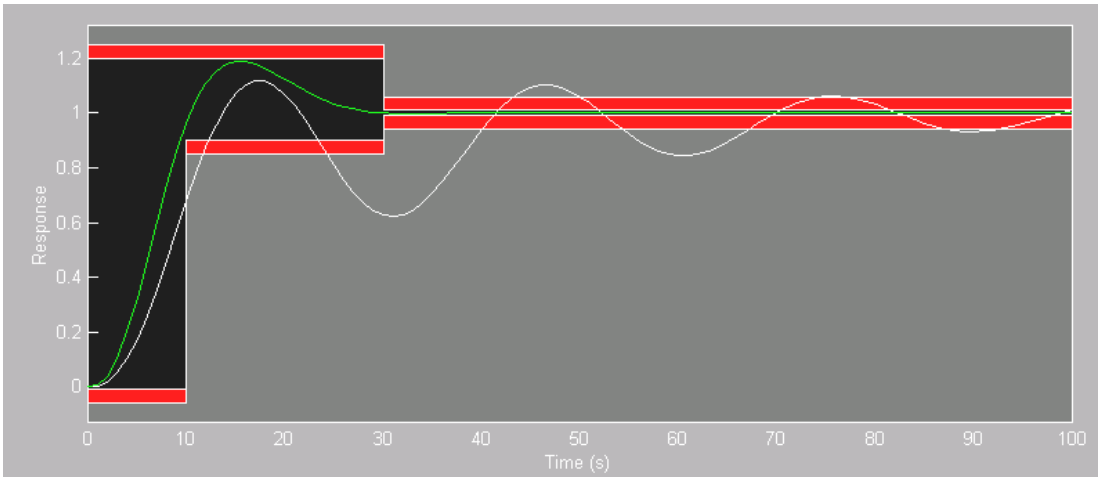
Parametry startowe regulatora wpisuje się w postaci symbolicznej w oknie właściwości bloku regulatora.



Rys. 4.7 Parametry startowe regulatora

W przestrzeni roboczej programu Matlab zadaje się ich wartości startowe. Następnie otwierany jest interfejs bloku NCD Outport (podwójne kliknięcie na bloku) w celu ustalenia wszystkich parametrów syntezy regulatora. W otwartym oknie widoczne są poziome i pionowe paski ograniczeń nałożonych na optymalizowaną charakterystykę odpowiedzi skokowej, które można za pomocą myszy dowolnie ustalać. W celu dokładnej edycji położenia paska ograniczeń możliwe jest określenie jego współrzędnych w oknie podręcznym otwiera-

nym po kliknięciu prawym przyciskiem myszy na obszarze paska. Na rysunku poniżej przedstawiono ograniczenia nałożone na optymalizowaną charakterystykę.



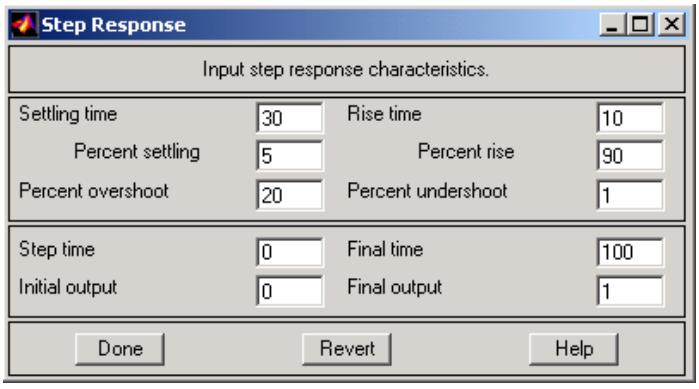
Rys. 4.8 Fragment interfejsu bloku NCD Outport z widocznymi ograniczeniami nakładanymi przez użytkownika na optymalizowany przebieg

Przed rozpoczęciem optymalizacji należy ustalić podstawowe parametry jej przebiegu. Po uruchomieniu opcji **Optimization | Parameters** otwiera się okno dostosowania parametrów o następującym znaczeniu:

Tunable Variables – zmienne decyzyjne, czyli poszukiwane parametry układu, zapisywane w postaci symbolicznej, poszczególne symbole należy oddzielić spacjami lub przecinkami.

Discretization interval – krok dyskretyzacji – im większy tym mniej punktów charakterystyki zostaje wyznaczanych, tym szybciej przebiega optymalizacja. W celu wygładzenia przebiegu optymalizowanej krzywej należy odpowiednio zmniejszyć tą wartość w zależności od całkowitego czasu symulacji. W przypadku optymalizacji przeprowadzanej dla dyskretnych modeli, wartość tego parametru może stanowić jedynie całkowite wielokrotności czasu próbkowania.

Variable Tolerance i **Constraint Tolerance** – dopuszczalny błąd zmiennych i ograniczeń, proponowane wartości 0.001.



Rys. 4.9 Okno wprowadzania zakresów optymalizowanej funkcji

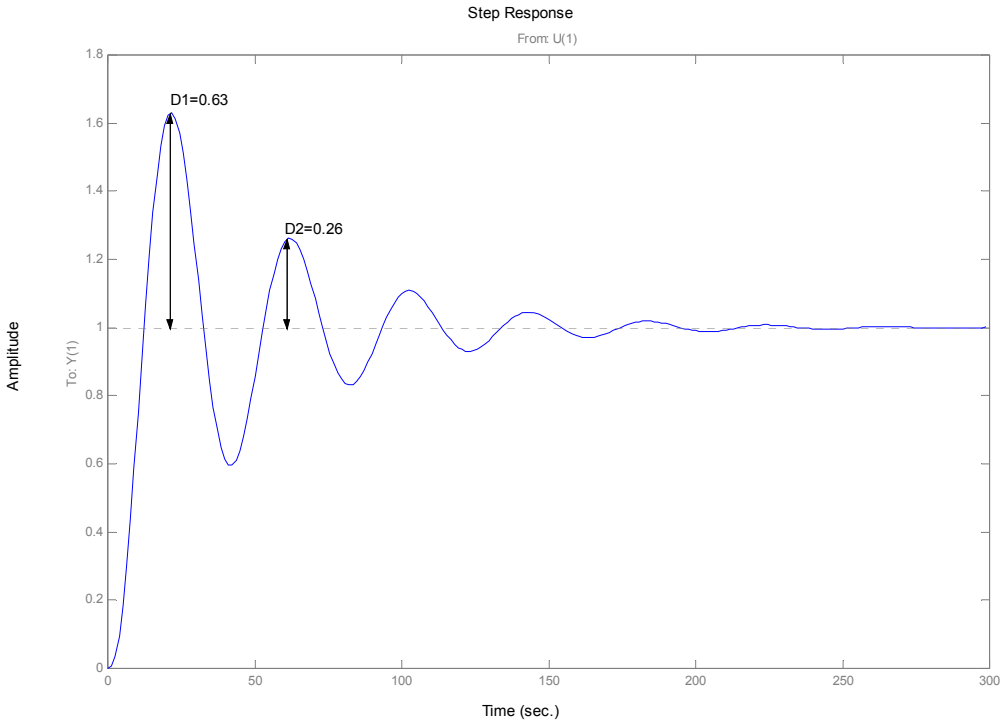
Powyższy rysunek przedstawia okno wprowadzania parametrów z podanymi wartościami dla omawianego przykładu. Po ustawieniu parametrów uruchamiana jest opcja **Options | Initial response,** inicjując początkową postać optymalizowanej charakterystyki.

Podstawową zaletą omawianego podejścia do syntezy regulatora PID jest nieporównywalnie mniejszy nakład pracy w stosunku do metod tradycyjnych oraz fakt, iż metoda ta nie wymaga od projektanta układu regulacji właściwie żadnego przygotowania teoretycznego w zakresie syntezy regulatora. Warunkiem zastosowania tej metody jest, obok posiadania odpowiedniego oprogramowania, znajomość modelu matematycznego rozpatrywanego układu. W przypadku modeli dyskretnych metodyka doboru parametrów regulatora jest podobna (należy zwrócić uwagę na prawidłową wartość parametru *Discretization interval*).

4.3 Metoda Zieglera - Nicholsa

Jednym z najbardziej istotnych problemów regulacji jest optymalne nastawianie parametrów regulatora dla określonego procesu i uzyskanie w ten sposób prawidłowego działania układu. Nastawy regulatorów ciągłych wyznacza się różnymi metodami na podstawie odpowiednich wzorów obliczeniowych czy tabel zapewniających minimalizację określonego wskaźnika jakości regulacji. Ziegler i Nichols poszukiwali takich nastaw regulatorów, które minimalizowałyby kryterium jakości regulacji IAE, trzyskokowej zmianie wartości zadanej. [51] Kryterium temu odpowiada w przybliżeniu wartość ilorazu dwóch pierwszych przeregulowań w układzie regulacji równa $D_2/D_1=30\div50$. Dla poniższego przypadku stosunek ten wynosi 41.27%

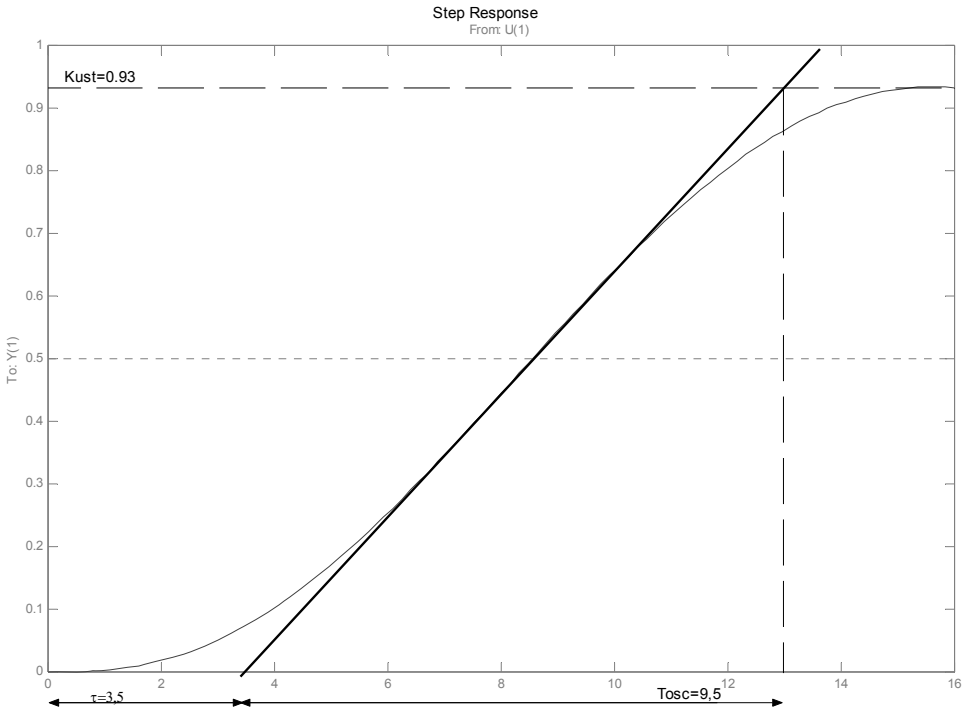
[51] Ziegler, J.G. and Nichols, N.B. *Optimum settings for automatic controllers circuit*. Transaction of the ASME, 1942 November, pages 759–768



Rys. 4.10 Parametry odpowiedzi jednostkowej układu regulacji

W zależności od możliwości wykonania eksperymentu identyfikacyjnego na obiekcie istnieją dwie metody postępowania przy określeniu nastaw regulatora: na postawie odpowiedzi obiektu regulacji na skok jednostkowy oraz na podstawie próby stabilności przeprowadzonej w układzie regulacji.

W metodzie pierwszej rejestruje się odpowiedź obiektu regulacji na skok jednostkowy. Niezależnie od charakteru odpowiedzi obiektu (obiekt statyczny, czy astatyczny) na skok jednostkowy wykreśla się styczną w punkcie o największym nachyleniu i wyznacza stałą czasu opóźnienia oraz unormowany wartością skoku jednostkowego wymuszenia tangens kąta nachylenia stycznej $R= tg(\alpha)/u$. Na podstawie parametrów T_0 i R ustala się nastawy regulatorów zgodnie z poniższa tabelą (próba skokowa)



Rys. 4.11 Odpowiedź jednostkowa obiektu regulacji

W drugiej metodzie próbę przeprowadza się w układzie regulacji z nastawionym regulatorem na działanie proporcjonalne. Zwiększa się wzmocnienie regulatora stopniowo do momentu, gdy w układzie wystąpią drgania niegasnące (granica stabilności), odczytuje się wówczas wzmocnienie regulatora K_{kr} oraz okres oscylacji T_{osc} w układzie i na ich podstawie wyznacz się parametry nastaw regulatora zgodnie z poniższą tabelą (próba oscylacyjna)

Typ regulatora	Wartości nastaw regulatora					
	Próba skokowa (R T_0)			Próba oscylacyjna (K_{kr} T_{osc})		
	Kp	***Ti***	***Td***	***Kp***	***Ti***	***Td***
P	$1/RT_0$			$0.5K_{kr}$		
PI	$0.9/R*T_0$	$3.3/T_0$		$0.45K_{kr}$	$0.83T_{osc}$	
PID	$1.2/R*T_0$	$2/T_0$	$0.5/T_0$	$0.6K_{kr}$	$0.5T_{osc}$	$0.125T_{osc}$

Tabela 4.1 Nastawy regulatorów według zasad Zieglera - Nicholsa[52]

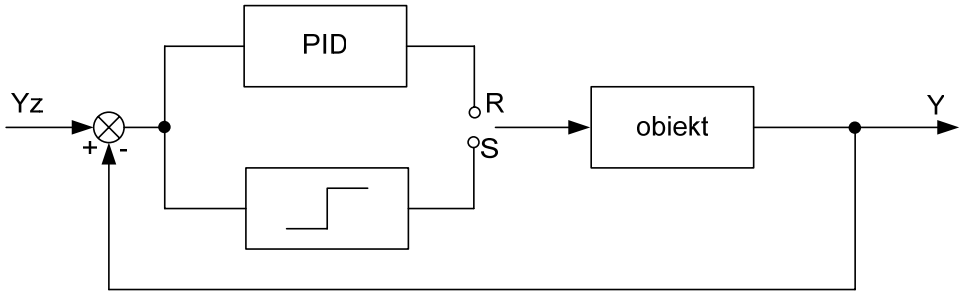
Inne kryterium doboru nastaw regulatorów analogowych zastosował Passen, w tzw. regule Passena, wykorzystując aperiodyczność odpowiedzi skokowej i minimalizację czasu regulacji. Autor metody wyprowadził następujące zależności określające wartości nastaw:

$$K_p = 0.2K_{kr};\ T_i = 0.33T_{osc};\ T_d = 0.5T_{osc}.$$

[52] *Laboratorium podstaw automatyki*, wydawnictwo uczelniane WSM, 1995r.

Propozycje Zieglera-Niecholsa i Passena dotyczące doprowadzenia obiektu do oscylacji i utrzymania amplitudy drgań na stałym poziomie są trudne do wykonania w sposób automatyczny. Poza tym nie wszystkie obiekty można doprowadzić do granic stabilności. W celu realizacji automatycznego wymuszenia oscylacji na stałym poziomie i określenia punktu krytycznego regulacji została opracowana metoda eksperymentu z przekaźnikiem. Metoda ta oparta jest na obserwacji, że układ z opóźnieniem fazowym, wynoszącym co najmniej -180 stopni, może posiadać przebieg oscylacyjny o okresie T_{osc}, pod kontrolą przekaźnika. Nastawy regulatora wyznacza się na podstawie wzorów podanych w powyższej tabeli przy próbie oscylacyjnej.

W celu określenia wzmocnienia krytycznego i okresu oscylacji, został wykorzystany układ jak na poniższym rysunku. Uchyb regulacji e jest sygnałem okresowym o okresie T_{osc}.



Rys. 4.12 Układ do automatycznego strojenia regulatora metodą eksperymentu z przekaźnikiem (położenie S) i regulacji PID (położenie R)

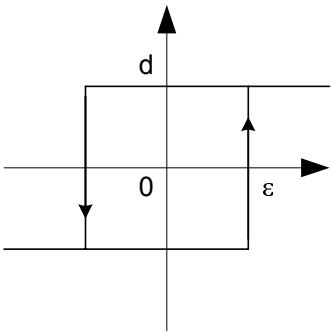
Jeśli *d* jest amplitudą skoku przekaźnika, to można z rozwinięcia szeregu Fourier'a wykazać, że pierwsza harmoniczna wyjścia przekaźnika posiada amplitudę $4d/\pi$. Jeśli a jest zmierzoną amplitudą szczytu błędu to wzmocnienie krytyczne wynosi w przybliżeniu:

$$K_{kr} = 4d/\pi a$$

Funkcja opisująca dla idealnego przekaźnika może być przybliżona wzorem:

$$N(a) = 4d/\pi a$$

Zamiast przekaźnika może być użyty inny układ nieliniowy, jakkolwiek przekaźnik z histerezą jest układem mniej czułym na szumy. Przy eksperymencie z przekaźnikiem, wartość histerezy przekaźnika ε określa się w zależności od modułu szumu



Rys. 4.13 Charakterystyka przekaźnika z histerezą

5 Badanie programu identyfikującego oraz poszukującego nastawy regulatora PID

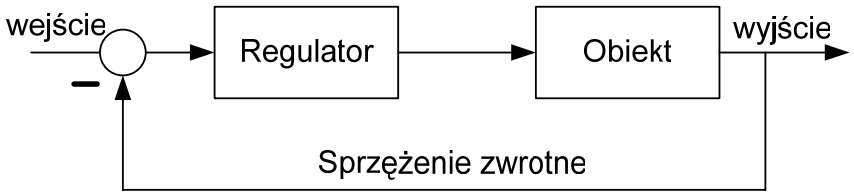
5.1 Identyfikacja obiektu do regulacji w układzie zamkniętym

Często interesuje nas dopasowanie prostego systemu niskiego rzędu do zmierzonej odpowiedzi skokowej systemu. Podejście to obrazuje analizowany w tym rozdziale obiekt inercyjny pierwszego rzędu opisywany modelem transmisyjnym.

$$Y(s)=G(s)*U(s)$$

gdzie *Y(s)* jest transformatą Laplace'a sygnału wyjściowego, *U(s)* jest transformatą Laplace'a sygnału wejściowego, a *G(s)* jest transformatą systemu.

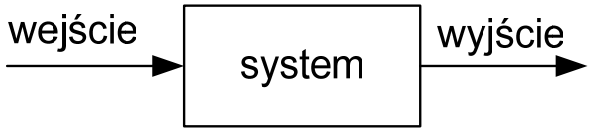
Na poniższym schemacie widzimy typowy układ ze sprzężeniem zwrotnym. Regulator został umieszczony w torze głównym. Spotyka się również rozwiązania z regulatorem w sprzężeniu zwrotnym.[53]



Rys. 5.1a Układ w pętli zamkniętej

5.1.1 Identyfikacja obiektu i filtracja danych

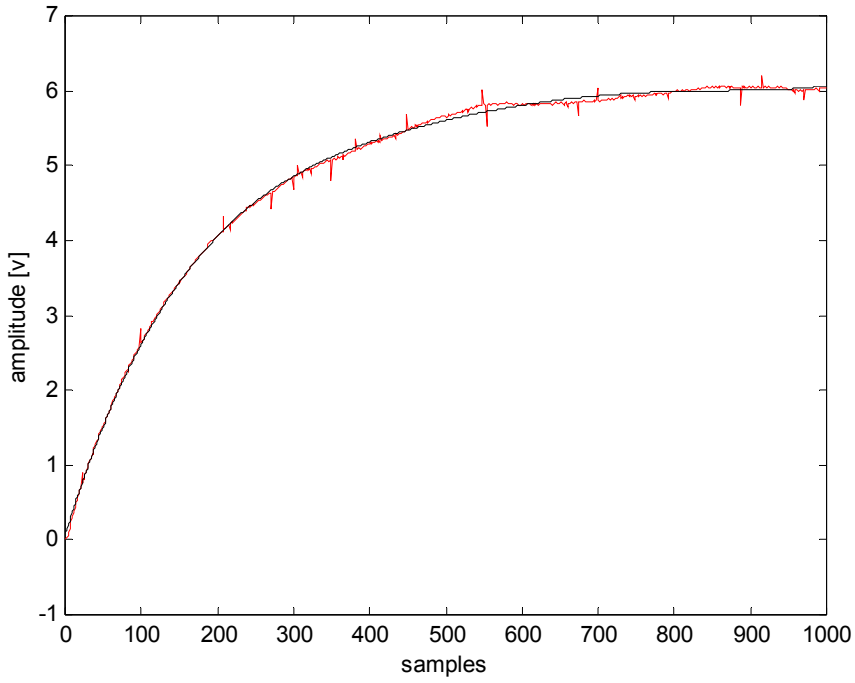
Układ identyfikowano w pętli otwartej tak jak to przedstawia poniższy rysunek.



Rys. 5.1b Układ w pętli otwartej

[53] patrz rys 2.3

Na wejście układu (zbiornik) podano skok jednostkowy, czyli pompa pracowała ze stałą wydajnością. Odpowiedzią układu jest poziom cieczy w zbiorniku, który dąży do stanu ustalonego, jak to można zaobserwować na poniższym wykresie.



Rys. 5.2 Odpowiedź układu na skok jednostkowy (czerwony), po aproksymacji wielomianem (czarny)

Na powyższym wykresie widać dwie charakterystyki. Pierwsza z nich została oznaczona kolorem czerwonym. Jest to odpowiedź układu na skok jednostkowy. Dane zostały wprowadzone do MATLABa za pomocą karty *Data Acquisition Unit* – AD512 produkowaną przez czeską firmę *Humosoft*. Jak można zauważyć dane są zakłócone. W celu usunięcia szumu, zaproksymowano funkcję do wielomianu piątego stopnia (*ang. Polynomial Curve Fitting*[54]) Fragment kodu programu usuwającego szum został zamieszczony poniżej.

```
x=[1:SamplingPeriod:N];
p = polyfit(x,Vout,5);        %Vout, zmienna zawierająca szum (kolor czerwony na powyższym
                              rysunku)
ypoly = polyval(p,x);         %ypoly, zmienna po aproksymacji wielomianem (czarny)
```

Piąty stopień wielomianu dawał wystarczająco dobre przybliżenie. Błąd względny dla filtracji wynosi:

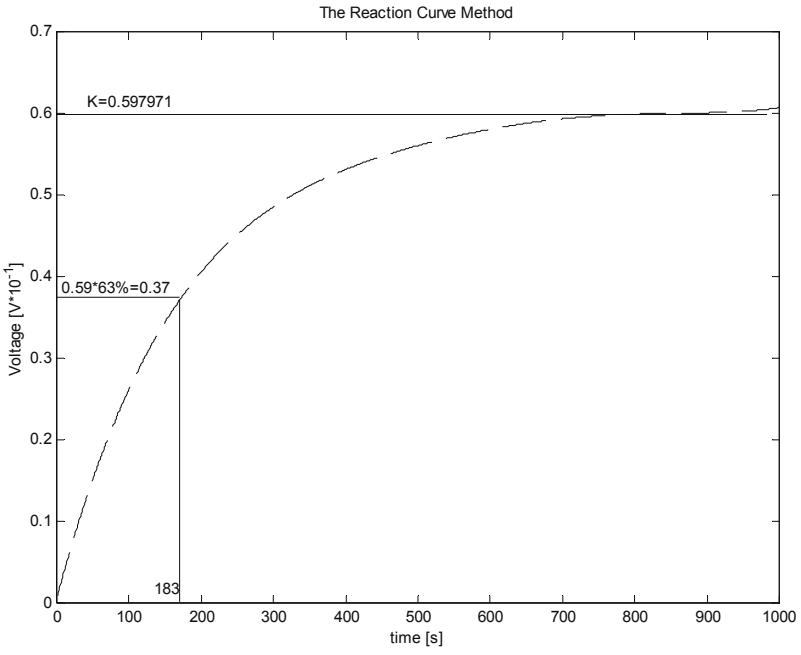
$$\delta = \frac{|\mathrm{Re} - aprox|}{\mathrm{Re}} = 3.0422\%$$

[54] więcej na temat tej metody w Internecie na oficjalnej stronie producenta MATLABa http://www.mathworks.com/access/helpdesk/help/techdoc/ref/polyfit.shtml

5.1.2 Metoda klasyczna (*The Reaction Curve Method*)[55]

Po zdefiniowaniu struktury modelu następnym krokiem jest dobór właściwych wartości dla danych parametrów. W celu zidentyfikowaniu tych parametrów musimy przeprowadzić odpowiednią symulację. W metodzie klasycznej (*ang. The reaction curve method*) na wejście układu podajemy skok jednostkowy i rejestrujemy odpowiedź układu. Następnie współczynniki obliczamy na podstawie otrzymanego wykresu.[56] Przykład zastosowania metody klasycznej do identyfikacji układu z opóźnieniem transportowym jest zamieszczony w książce Torstena Söderström oraz Petre Stoica, *System Identification*, wydaną przez *Prentice Hall International w* 1994r. Podczas identyfikacji korzystano z pomocy konsultanta Toma O'Mahonego oraz jego publikacji *Real–time Control of a Single–tank System using Humusoft* Electronics Dept., CIT 2000. oraz *Real–time fluid level control using PID and GPC,* Proc. First International Postgraduate Research Students Conf. DIT Dublin 1998.[57]

Odpowiedź układu na skok jednostkowy jest pokazana poniżej.



Rys. 5.3 Metoda klasyczna (ang. The reaction curve method)

Rezultatem identyfikacji jest model matematyczny obiektu inercyjnego pierwszego rzędu.

[55] obliczenia w pliku `ident_calc.m`,

[56] E.F. Camacho and C. Bordons, *Model Predictive control in the process industry,* Prentice Hall International

[57] współautorem tej publikacji jest C. J. Downing

$$K(s) = \frac{0.597971}{183s + 1}$$

$$minera\,l^{58}\ K(s) = \frac{0.003268}{\mathrm{s} + 0.005464}$$

Błąd względny dla tej metody wynosi $\delta = \frac{|\mathrm{Re} - estim|}{\mathrm{Re}} = 1.9377[\%]$

Metoda klasyczna jest prawdopodobnie jedną z najbardziej popularnych metod używanych w przemyśle do identyfikacji obiektów. Fragment kodu programu zaimplementowanego do środowiska Matlab został zamieszczony poniżej.

```
if IdentMethods==1            %identification using classical method
k=(ypoly(N)-ypoly(1))/1; %(u(N)-u(1));
stepchange=(1-exp(-1))*k;     % 63% of settling value
final_value=(stepchange+ypoly(1));
for i=1:N;                            %find the first sample
   if ypoly(i) <= final_value;
      ts=x(i);
   end
end
disp 'plant computes using a classical method'
PlantS=tf([k],[ts 1]) % transfer function of a plant
end;
```

Całość kodu została zamieszczona w załącznikach, w pliku pod nazwą `ident_calc.m`

5.1.3 Metoda Box–Jenkins

Ogólny liniowy model z wejściem *u* i wyjściem *y* może zostać zapisany jako:

$$A(q)y(t) = \sum_{i=1}^{nu}[\frac{B_i(q)}{F_i(q)}]u_i(t - nk_1) + [\frac{C(q)}{D(q)}]e(t)$$

gdzie u_i oznacza wejście, natomiast *i*–numer. $A,\ B_i,C,D$ oraz F_i są współczynnikami wielomianu (z lub q)

[58] polecenie `minreal` w Matlabie, przekształca transformatę w ten sposób, aby wyrugować wartość przy operatorze Laplace'a - „s"

Dla metody Box–Jenkins $y(t) = [\frac{B(q)}{F(q)}]u(t-nk) + [\frac{C(q)}{D(q)}]e(t)$

Poniżej przedstawiono fragment kodu wykorzystującego metodę Box–Jenkins.[59]

```
elseif IdentMethods==2
%Computes the prediction error estimate of a Box-Jenkins model.
th=bj(z,[1 0 0 1 0]); % z-meassured data
[Num,Den]=th2tf(th);% transforms from the THETA-format to transfer functions.
disp 'Plant computes using a Box-Jenkins model'
plantbj=tf(Num,Den,1);
PlantS=d2c(plantbj,'zoh')     % conversion of discrete LTI models to continuous time.
                                    % 'zoh' Assumes zero-order hold on the inputs.
```

Funkcja przejścia (*ang. Transfer function*) badanego obiektu:

$$K(s) = \frac{0.003298\,\mathrm{s} + 0.003307}{\mathrm{s} + 0.005471}$$

$$\textit{Błąd względny}\ \delta = \frac{|\mathrm{Re} - estim|}{\mathrm{Re}} = 0.9502[\%]$$

Porównano metodę *Box–Jenkins*[60] *z* innymi metodami identyfikacyjnymi. Udowodniono, iż ta metoda ma najniższy błąd względny. Podczas studiów w *Cork Institute of Technology* porównano metody klasyczną, najmniejszych kwadratów oraz Box–Jenkins do identyfikacji silnika prądu stałego i wysunięto wniosek, że jest to najlepsza metoda identyfikacyjna. Moje obliczenia to potwierdziły.

5.1.4 Metoda najmniejszych kwadratów

W niniejszym podrozdziale omawia się i analizuje się podstawy regresji liniowej. Jest to bardzo rozpowszechniona metoda statystyczna, której pierwotną wersję można spotkać już w pracach Gaussa, poświęconych obliczaniu orbit planetarnych.[61]

W regresji liniowej stosuje się najprostszy typ modelu parametrycznego. Struktura tego modelu ma postać

---

[59] obliczenia w pliku `ident_calc.m`

[60] szerzej G.N. Polydoras, J. S. Anagnopoulos, G. Ch Bergels, *Air quality predictions: dispersion model vs Box-Jenkins stochatistic models. An implementation and comparison*, Applied Thermal Engineering 18 (1998), pages 1037-1048

[61] Gauss K. F., *Teoria Motus Corporum Coelestium in Sectionibus Conicus Solem Ambientum* 1809 (tłumaczenie: *Theory of motion of the heavenly bodies moving about the sun in conic section*, Dover, New York)

$$y(t) = \varphi^T(t)\theta$$

gdzie *y(t)* jest wielkością mierzoną, φ(t) jest *n* – wymiarowym wektorem wielkości znanych, zaś θ jest *n*–wymiarowym wektorem nieznanych parametrów. Współrzędne wektora *φ(t)* są często nazywane *zmiennymi regresji*, a wielkość *y(t) zmienną wyjściową*. Wektor Q nazywamy *wektorem parametrów*. O zmiennej *t* zakładamy, że przybiera wartości całkowite. Może się zdarzyć, że *t* oznacza czas, ale nie jest to regułą.

Model ten można z łatwością uogólnić na przypadek wielowymiarowy, dla którego

$$y(t) = \Phi^T(t)\theta$$

gdzie *y(t)* i *θ* są wektorami odpowiednio nieznanych - i n–wymiarowymi, natomiast *Φ(t)* jest macierzą (n | p)–wymiarową.[62].

$$\textit{Dla modelu ARX}^{63}\quad A(q)y(t) = B(q)u(t-nk) + e(t)$$

Poniżej przedstawiono fragment kodu identyfikującego metodą najmniejszych kwadratów.[64]

```
elseif IdentMethods==3
%Computes LS-estimates of ARX-models.
disp 'plant computes using LS'
th=arx(z,[1 1 1]);           %  Computes LS-estimates of ARX-models.
[Num,Den]=th2tf(th);
plantarx=tf(Num,Den,1);
```

$$\textit{Transformata przejścia}\quad K(s) = \frac{0.003327}{s + 0.005505}$$

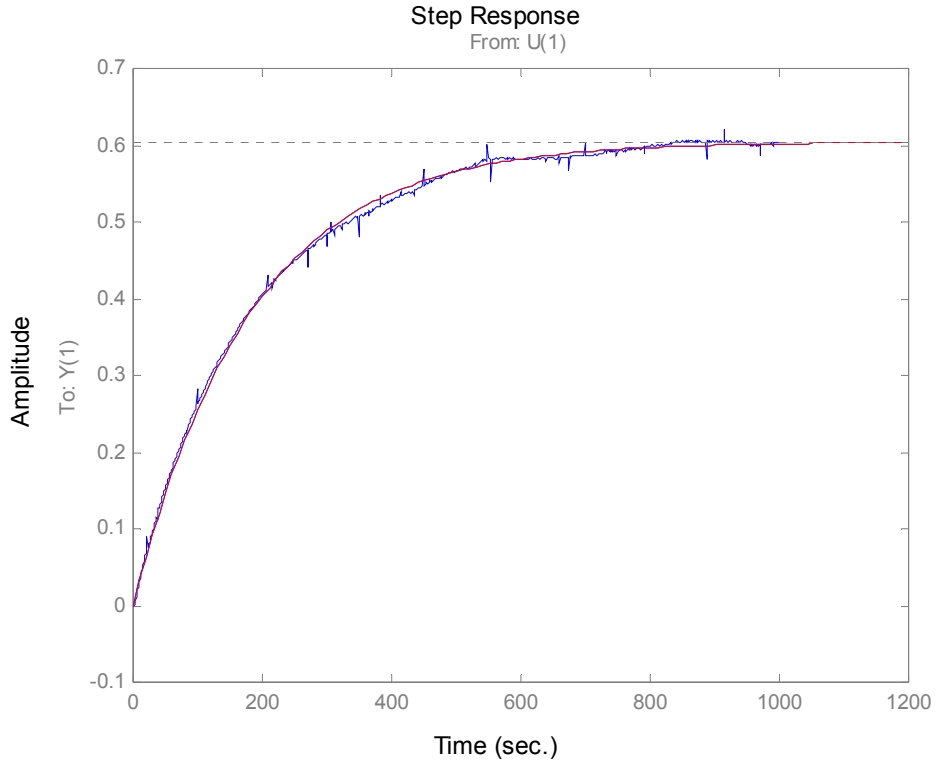
Błąd względny dla metody najmniejszych kwadratów

$$\delta = \frac{|\mathrm{Re} - estim|}{\mathrm{Re}} = 1.2829[\%]$$

[62] więcej na temat estymacji metodą najmniejszych kwadratów dla parametrów modelu wielowymiarowego w uzupełnieniu C7.3, Torsten Söderström, Pstre Stoica, *System Identification*, Prentice Hall International 1994r.

[63] szerzej o identyfikacji za pomocą Matlaba w podręczniku użytkownika, *System Identification* s. 50

[64] całość kodu została zamieszczona w załączniku A w pliku pod nazwą `ident_calc.m`



Rys. 5.4 Porównanie modelu matematycznego zbiornika (czerwony), z danymi rzeczywistymi (niebieski)

Podsumowując przeanalizowane metody identyfikacyjne, można napisać iż zwycięzcą jest Box – Jenkins. Dla tej metody błąd względy jest najmniejszy. Dobra praktyka inżynierska sugeruje porównanie modelu matematycznego z fizyczną odpowiedzią układu.[65] Taka analiza została przeprowadzona i zamieszczona na powyższym wykresie. Porównanie można również dokonać za pomocą programu opisanego w rozdziale 5.6. Podczas dokonywania porównań użyteczna jest funkcja HOLD ON.

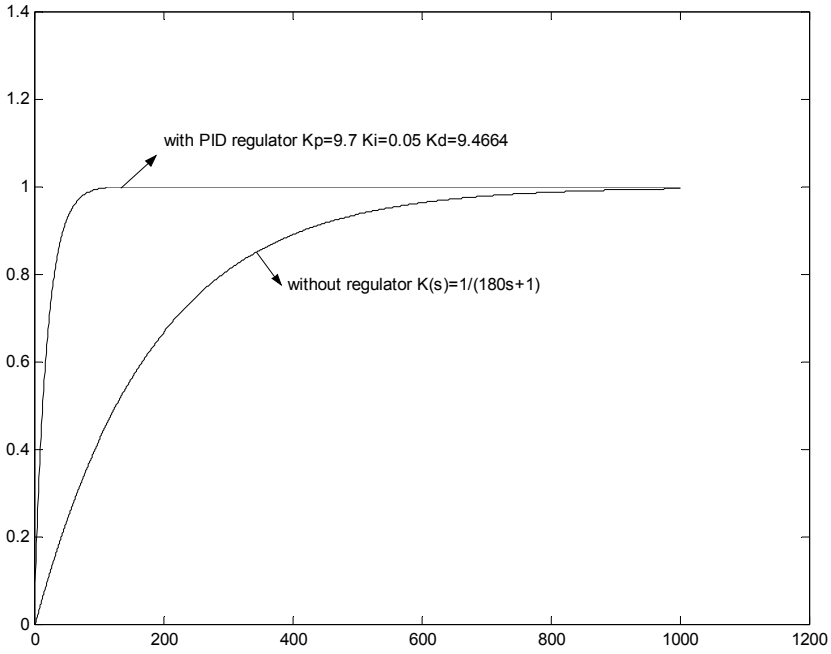
5.2 Rezultat strojenia regulatora PID przy wykorzystaniu AG

5.2.1 System inercyjny pierwszego rzędu

Poniżej przeprowadzono dobór nastaw regulatora PI(D) dla obiektu inercyjnego pierwszego rzędu takiego jak zbiornik. Transmitancja obiektu $K(s) = \frac{1}{180s+1}$

[65] Astrom, K. J., Eykoff, P., *System Identification – a survay. Automatica, 7,* 1971r., s. 123-167

Czas obliczeń w obu przypadkach (PI oraz PID) około 30 [s][66]. Około jednej sekundy na jedną iterację.

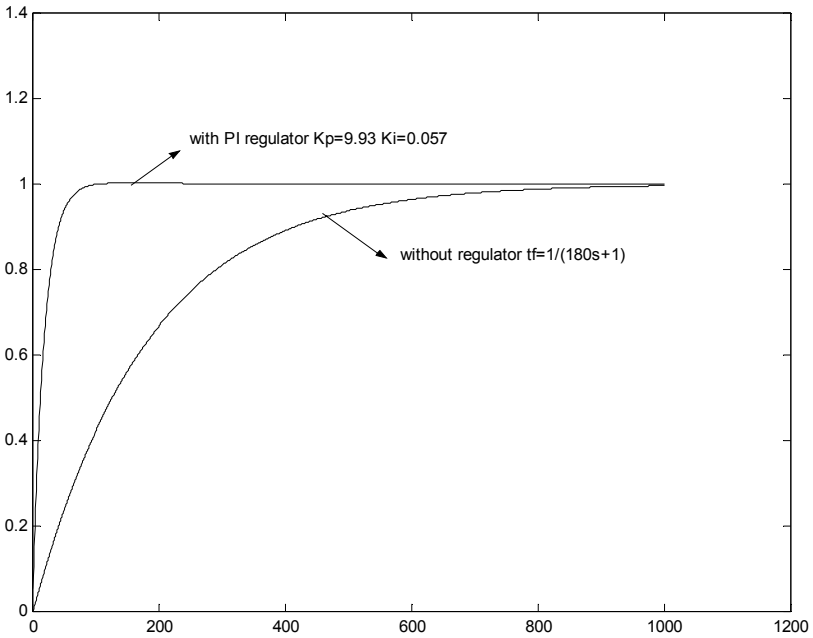


Rys. 5.5 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PID otrzymane za pomocą AG

Dla powyższej odpowiedzi na skok jednostkowy otrzymano następujące kryteria całkowe:

IAE=18.2355 ISE=8.3939 ITAE= 433.2107

[66] Wszystkie obliczenia były przeprowadzane na komputerze (Pentium III 700 MHz.) o mocy obliczeniowej 1880 MIPS (milionów operacji na sekundę) oraz 935 MFLOPS (operacji zmiennoprzecinkowych).

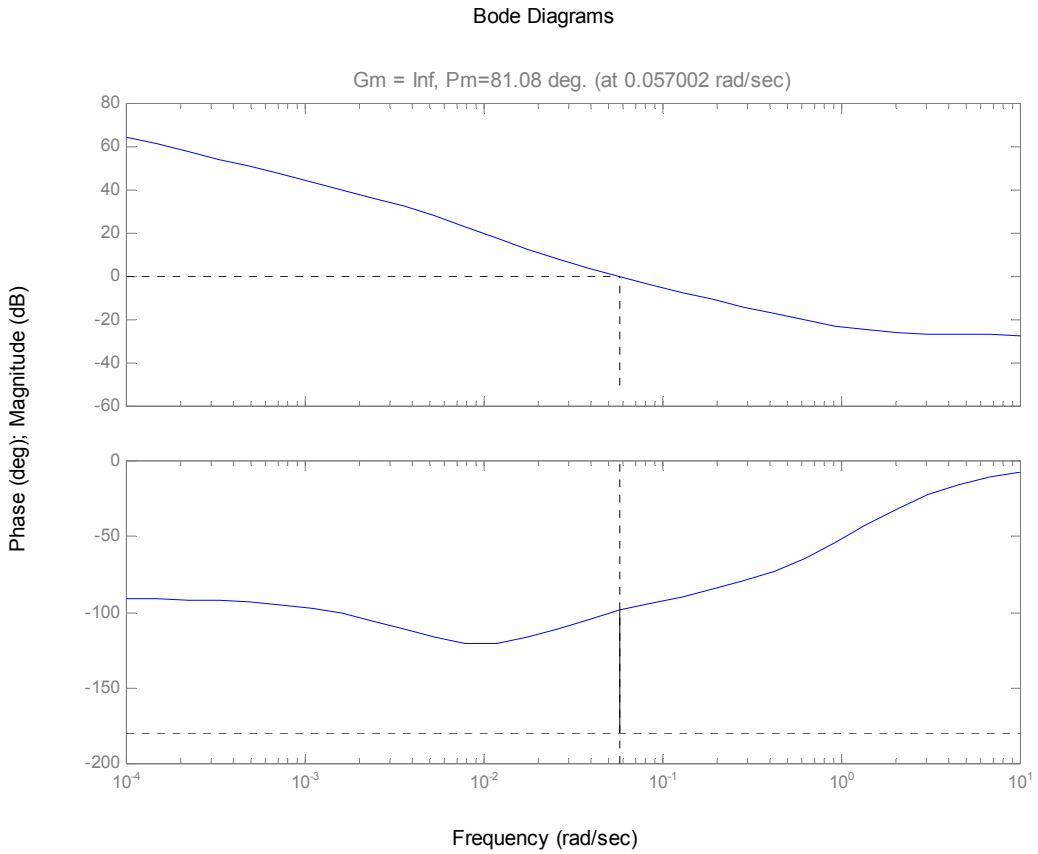


Rys. 5.6 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PI otrzymane za pomocą AG

Kryteria całkowe:

IAE=17.7906 ISE= 8.5279 ITAE= 447.0941

Kryteria całkowe dla tego obiektu można zmniejszyć poprzez zmianę zakresów Kp Ki Kd. W programie przyjęto ograniczenia Kp Ki Kd w zakresie 0÷10. Jak widać na poniższych charakterystykach Bodego, obiekt z regulatorem posiada nieskończony zapas modułu. Wniosek stąd taki, że współczynnik wzmocnienia dla tego typu obiektu może być zwiększony. W przypadku zwiększenia zakresu Kp wartości kryteriów całkowych będą mniejsze.



Rys. 5.7 Logarytmiczna charakterystyka amplitudowa i fazowa w układzie otwartym z regulatorem PI

Dla powyższego układu zapas modułu dąży do nieskończoności. Z tego też powodu zastosowanie próby oscylacyjnej metody Zieglera–Nicholsa[67] dla tego typu obiektu jest niemożliwe.

5.2.2 System inercyjny pierwszego rzędu, strojenie bez wartości startowych

Algorytmy genetyczne mogą zakończyć prowadzić obliczenia po osiągnięciu satysfakcjonującego nas minimum funkcji kosztów bądź po przeprowadzeniu ustalonej liczby iteracji. Można również przyjąć oba założenia. W poniższym przykładzie założono, że algorytm genetyczny powinien prowadzić obliczenia nie dłużej niż 100 iteracji, co odpowiada około 100 sekundom.

[67] Ziegler, J.G. and Nichols, N.B. (1942) *Optimum settings for automatic controllers circuit*. Transaction of the ASME, November 759-768

Założenia są definiowane w pliku `genhaploid.m`[68] Fragment tego kodu został zamieszczony poniżej:

```
options = foptions([1 1e-3]);        %count up to 1e-3
options(14) = 100;                   %The default maximum generations
```

L.p.	Ilość iteracji	Kp	Ki	Kd	IAE
Zakres strojenia		0÷10	0÷10	0÷10	
1	21	9.1850	0.050965	1.8074	19.13
2	66	9.9454	0.0569	1.8137	17.71
3	36	9.9544	0.05874	2.3857	17.8739
4	29	9.9919	9.6333	1.2004	20.3752
5	72	10.000	0.05859	1.5390	17.73
6	100	9.9203	0.055695	2.7842	17.69
7	8	9.9963	0.0594	2.8087	17.84
8	75	9.8356	0.0544	4.0153	17.87
9	27	9.9951	0.05965	4.0008	17.56
10	85	9.9887	0.05676	6.503	17.66
Średnia	51.9				18.143

Tabela 5.1 System inercyjny pierwszego rzędu, strojenie regulatora PID za pomocą AG bez początkowej populacji

W szóstym przykładzie można zaobserwować zrealizowanie założenia zatrzymania pracy algorytmu genetycznego po 100 iteracjach.

5.2.3 System inercyjny pierwszego rzędu, AG z wartościami startowymi

W tym podrozdziale rozważony jest przykład z wartościami początkowymi. Innymi słowy w pierwszej populacji podano rozwiązanie.

```
options = foptions([1 1e-3]); %count up to 1e-3
options(14) = 100;                                  %The default maximum generations
x0=[4.0008 9.9951 0.05965];          % Kd Kp Ki
```

L.p.	Ilość iteracji	Kp	Ki	Kd	IAE
Zakres strojenia		0÷10	0÷10	0÷10	
Wartość początkowa populacji		9.9951	0.05965	4.0008	17.56
1	38	10	0.0555	4.9966	17.56
2	53	9.9260	0.0569	4.2506	17.79
3	74	9.9968	0.0565	3.3396	17.58
4	15	9.9151	0.0597	4.0008	17.89
5	68	9.9626	0.0574	5.8843	17.76
6	19	9.9951	0.0586	2.1282	17.75
7	33	9.9826	0.0546	2.0430	17.81
8	75	9.9951	0.0597	4.0008	17.89

[68] szczegóły w załączniku A

9	11	9.9151	0.0589	0.9526	17.77
10	100	9.9484	0.0503	3.5195	17.85
Średnia	48.6				17.765

Tabela 5.2 System inercyjny pierwszego rzędu, strojenie regulatora PID za pomocą AG, z rozwiązaniem w pierwszej populacji

Różnica pomiędzy dwoma przypadkami (z rozwiązaniem w początkowej populacji i bez niej) nie jest znacząca. Wynosi zaledwie 6.79% więcej iteracji niż dla przypadku pierwszego. Natomiast średni błąd jest o 2.15% wyższy w przypadku drugim (z rozwiązaniem w pierwszej populacji).

5.2.4 System inercyjny pierwszego rzędu, szeroki zakres nastawy regulatora

W trzecim przypadku założono zakres strojenia regulatora w granicach 0÷10000.

L.p	Ilość iteracji	Kp	Ki	Kd	IAE
Zakres strojenia		0÷10000	0÷10000	0÷10000	
1	100	6787.2	50.812	1404.1	$8.3*10^{-3}$
2	18	3378.1	9780.8	123.75	$9.2*10^{-3}$
3	25	1985.8	9822.8	247.3	$5*10^{-3}$
4	100	7088.1	39.4	327.5	$34.79*10^{-6}$
5	100	8378.1	46.5	164.8	$14.66*10^{-12}$
6	8	4794.4	4965.3	4794.4	$12.4*10^{-3}$
7	100	9009.5	50	109.7	$1.86*10^{-9}$
8	100	9890.3	54.9	437.5	$52.5*10^{-9}$
9	100	8900.4	49.4	247	$2.29*10^{-9}$
10	100	2359.1	9952.5	82.7	$43.98*10^{-6}$

Tabela 5.3 System inercyjny pierwszego rzędu, strojenie regulatora PID za pomocą AG z szerokim zakresem nastaw

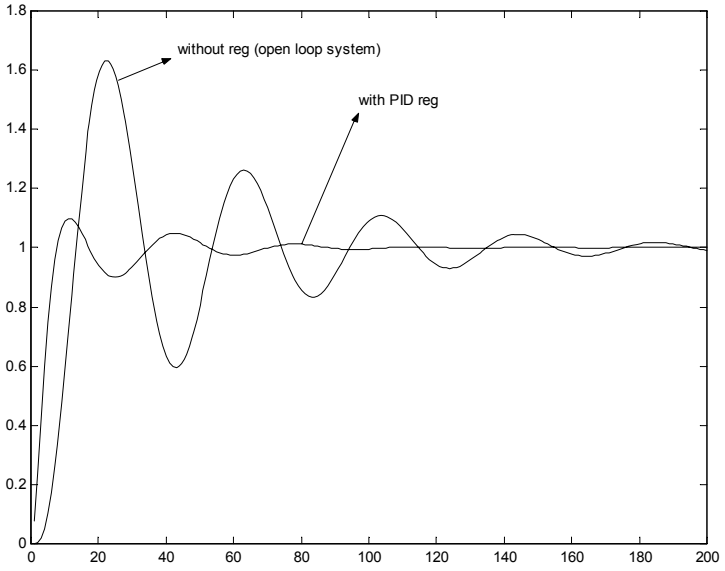
Znalezienie nastaw regulatora w tym przypadku zajęło więcej czasu. Kryterium błędu jest znacznie mniejsze niż dla regulatorów z zakresem 0–10. W przemyśle nie są stosowane regulatory z tak szerokim zakresem nastaw, więc jeśli nawet znajdziemy matematyczny model regulatora (współczynniki Kp Ki Kd), to zaimplementowanie go do obiektu rzeczywistego jest trudne do zrealizowania w regulatorach analogowych.

5.2.5 Obiekt oscylacyjny

W tym podrozdziale rozważono typowy obiekt oscylacyjny o następującej transmitancji:

$$K(s) = \frac{1}{50s^3 + 43s^2 + 3s + 1}$$

Jako funkcję przystosowania założono kryterium całkowe IAE.



Rys. 5.8 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PID otrzymane za pomocą AG, funkcja przystosowania obliczona na podstawie kryterium całkowego IAE

Współczynniki wzmocnienia regulatora PID:

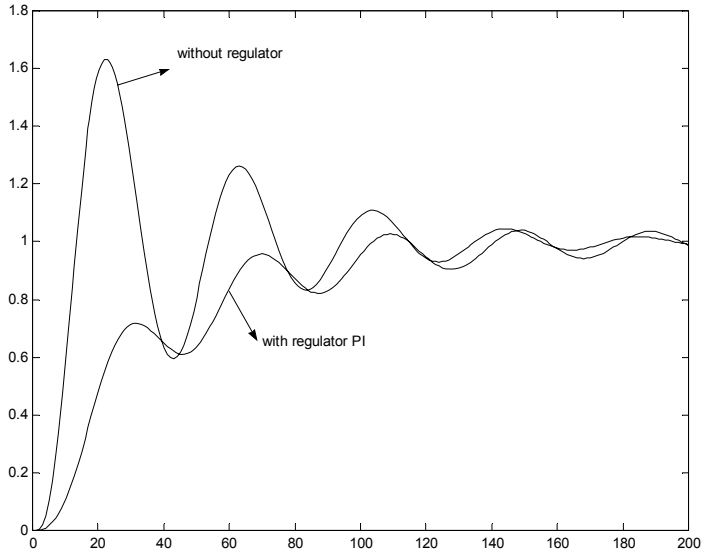
Kp=0.93935 Ki=0.2681 Kd=9.7963

Kryteria całkowe:

IAE=5.90 ISE=2.17 ITAE=110.02

Dla powyższego regulatora jako funkcję celu przyjęto kryterium całkowe IAE.

W poniższych trzech przykładach przeanalizowano poszukiwanie nastaw regulatora PID za pomocą algorytmów genetycznych dla trzech różnych funkcji przystosowania.



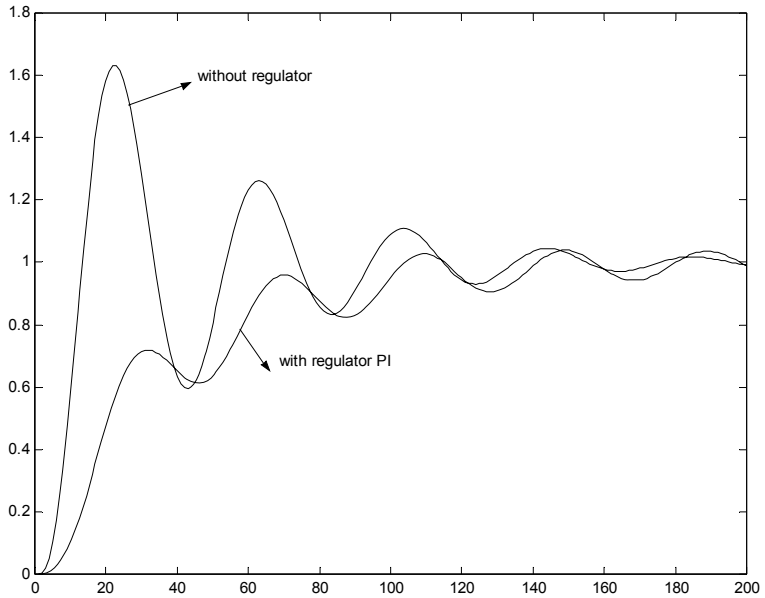
Rys. 5.9 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PI otrzymane za pomocą AG, funkcja przystosowania obliczona na podstawie kryterium całkowego IAE

Współczynniki wzmocnienia regulatora PI:

Kp=0.061 Ki=0.027

Kryteria całkowe:

IAE=40.74 ISE=19.5981 ITAE=2569



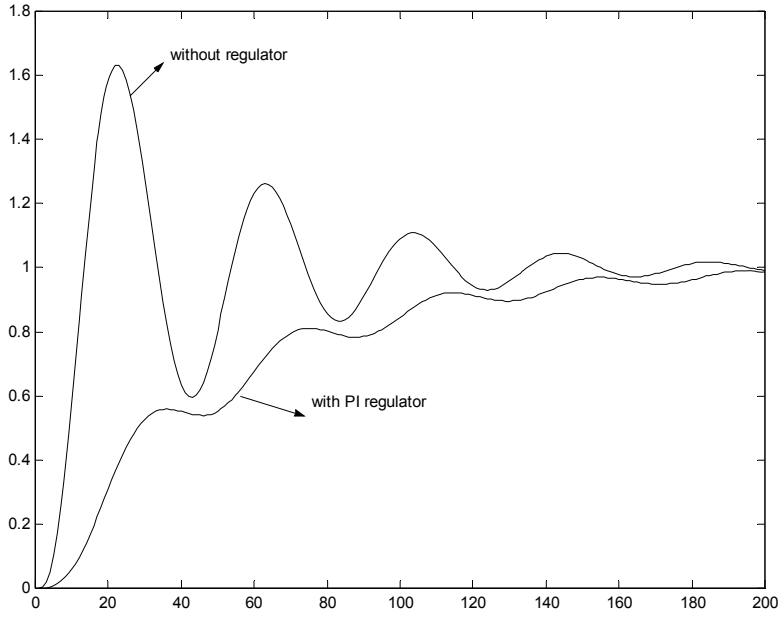
Rys. 5.10 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PI otrzymane za pomocą AG, funkcja przystosowania obliczona na podstawie kryterium całkowego ISE

Współczynniki wzmocnienia regulatora PI:

Kp=0.057984 Ki=0.027466

Kryteria całkowe:

IAE= 40.7471 ISE= 19.6519 ITAE=2656



Rys. 5.11 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PI otrzymane za pomocą AG, funkcja przystosowania obliczona na podstawie kryterium całkowego ITAE

Współczynniki wzmocnienia regulatora PI:

Kp=0.020752 Ki=0.019

Kryteria całkowe:

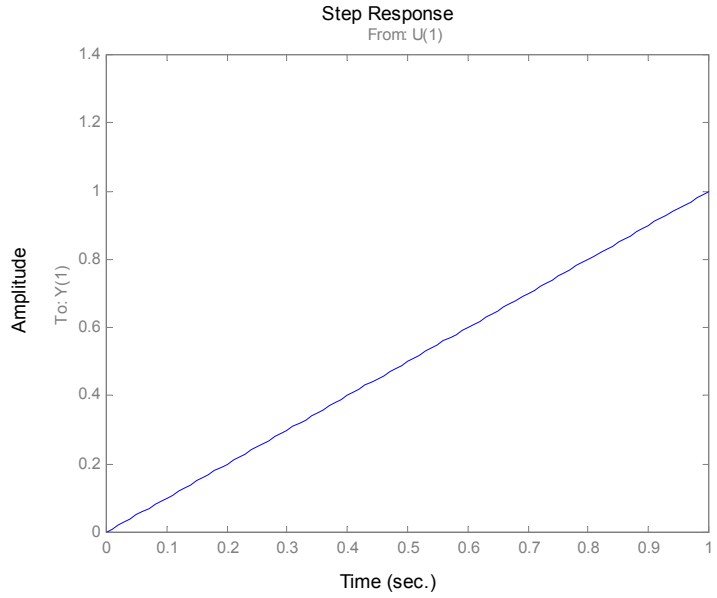
IAE= 51.9914 ISE= 27.3662 ITAE=2657

Dla obiektu przedstawionego na rys 4.10 jako funkcję przystosowania[69] (*ang. fitness function*) przyjęto kryterium całkowe IAE. Nie jest to znaczące, jakie kryterium całkowe przyjmiemy za funkcję przystosowania. Algorytmy genetyczne znajdują prawie to samo rozwiązanie – nastawy regulatora PID. Różnica pomiędzy wynikami istnieje ze względu na fakt, iż AG są metodą probabilistyczną.

[69] W literaturze można również spotkać inne nazwy takie jak: funkcja kosztów (*ang. cost function*), funkcja celu, funkcja minimalizowana bądź optymalizowana.

5.2.6 Obiekt niestabilny

Rozważmy dobór regulatora dla obiektu niestabilnego w pętli otwartej. Typowym obiektem jest człon całkujący. $K(s)=\frac{1}{s}$ W celu wprowadzenia tego typu obiektu do programu należy wpisać w polu zer [1], natomiast w polu biegunów [1 0].[70]



Rys. 5.12 Człon całkujący

Strojenie regulatora przeprowadzono za pomocą AG.

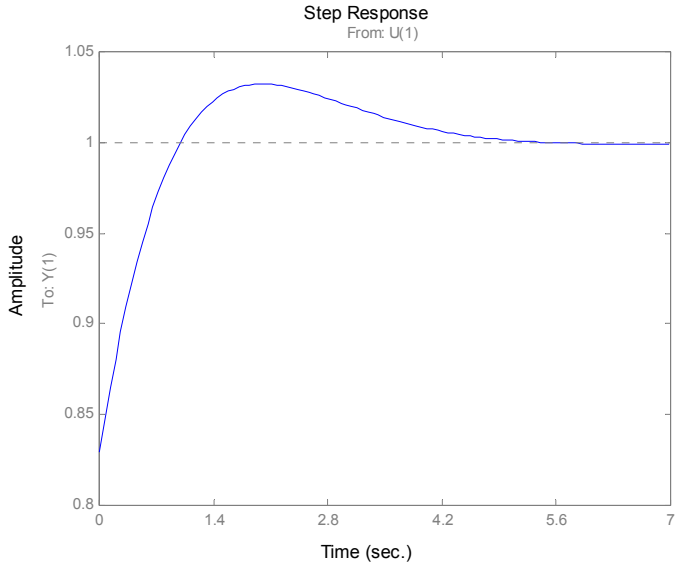
Współczynniki wzmocnienia regulatora PID:

Kp=9.8172 Ki=6.9795 Kd=4.85

Kryteria całkowe:

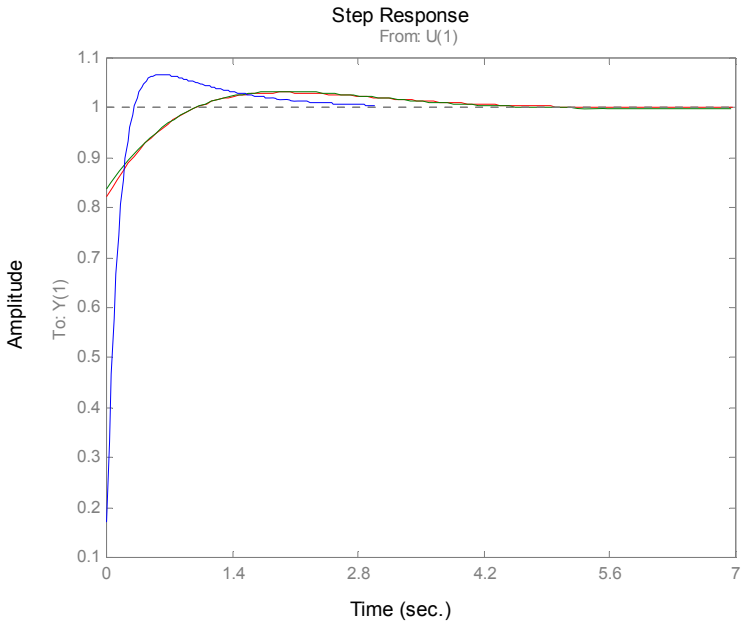
IAE= 0.065 ISE= 0.00155 ITAE=0.17958

[70] Patrz rozdział 5.6 *Graficzny interfejs użytkownika*



Rys. 5.13 Współczynniki regulatora PID otrzymane za pomocą AG, funkcja przystosowania obliczona na podstawie kryterium całkowego IAE

Porównanie metod optymalizacyjnych dla trzech różnych funkcji przystosowania



Rys. 5.14 Porównanie odpowiedzi układu na skok jednostkowy z regulatorem PID dla trzech różnych funkcji przystosowania IAE (czerwony) ISE (zielony) ITAE (niebieski)

Współczynniki wzmocnienia regulatora PID, funkcja przystosowania **IAE.**

Kp=9.9855 Ki=6.4415 Kd=4.5805

Kryteria całkowe:

IAE= 0.06451 ISE= 0.00145 ITAE=0.17841

Współczynniki wzmocnienia regulatora PID, funkcja przystosowania **ISE.**

Kp=9.99716 Ki=7.6817 Kd=5.2137

Kryteria całkowe:

IAE= 0.06378 **ISE= 0.00155** ITAE=0.17442

Współczynniki wzmocnienia regulatora PID, funkcja przystosowania **ITAE.**

Kp=9.8967 Ki=9.8712 Kd=0.20682

Kryteria całkowe:

IAE= 0.073727 ISE= 0.0028243 **ITAE=0.10769**

Na podstawie powyższego przykładu możemy zauważyć różnice w otrzymanych nastawach regulatorów. W ostatnim przypadku układ szybciej osiąga wartość zadaną i ma większe przeregulowanie[71]. Zależności pomiędzy funkcją celu, a kryterium całkowym zostały opisane w rozdziale 4.1.5

Współczynniki wzmocnienia regulatora PI, funkcja przystosowania **IAE.**

Kp=9.9973 Ki=9.7989

Kryteria całkowe:

IAE= 0.070199 ISE= 0.0024657 ITAE=0.10522

Współczynniki wzmocnienia regulatora PI, funkcja przystosowania **ISE.**

Kp=8.0093 Ki=0.0032044

Kryteria całkowe:

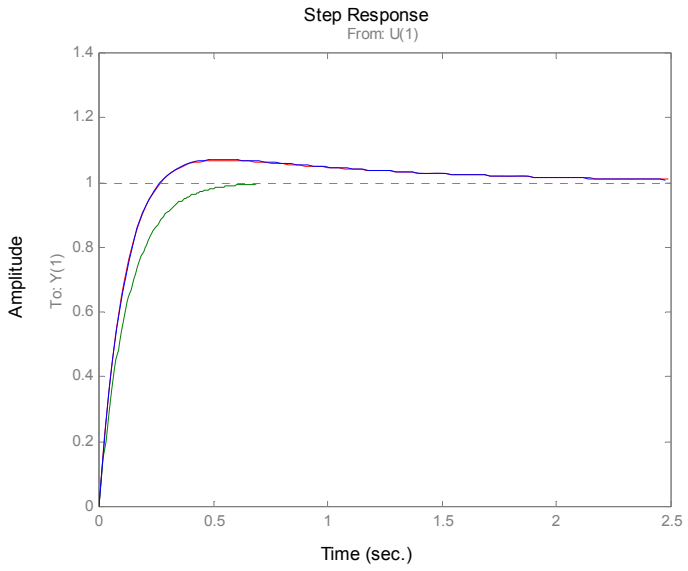
IAE= 0.041399 **ISE= $1.7946*10^{-6}$** ITAE=19.2352

Współczynniki wzmocnienia regulatora PI, funkcja przystosowania **ITAE.**

Kp=9.8791 Ki=9.9919

Kryteria całkowe:

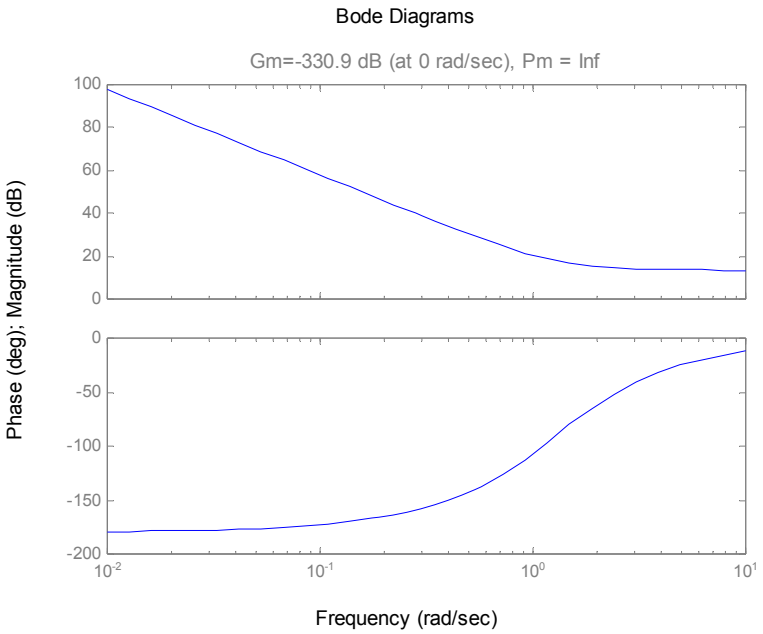
IAE= 0.070263 ISE= 0.002547 **ITAE=0.1032**



Rys. 5.15 Porównanie odpowiedzi układu na skok jednostkowy z regulatorem PI dla trzech różnych funkcji przystosowania IAE (czerwony) ISE (zielony) ITAE (niebieski)

[71] wartość przeregulowania w AG można kształtować wg potrzeb użytkownika w pliku `genpid.m`

Z powyższych danych można wywnioskować, iż kryterium całkowe jest najniższe dla przypadków, gdy dane kryterium stanowiło funkcję celu. Innymi słowy, jeśli funkcją celu było kryterium ITAE, to błąd ITAE dla tego przypadku jest względnie mniejszy niż dla IAE bądź ISE.



Rys. 5.16 Logarytmiczna charakterystyka amplitudowa i fazowa w układzie otwartym z regulatorem

Jak widać na powyższej charakterystyce, układ z regulatorem jest stabilny.

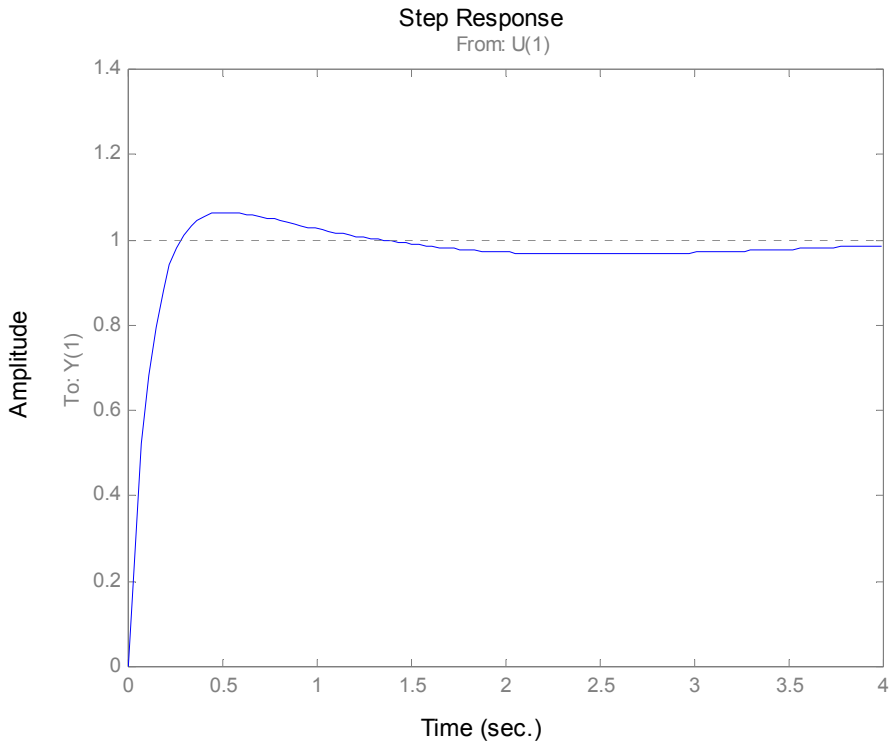
5.2.7 Obiekt na granicy stabilności

Rozważmy następującą transmitancję:

$$K(s) = \frac{1}{s^2 + 1}$$

Jest to przykład kuli umieszczonej na równoważni. Odpowiedź układu bez regulacji na skok jednostkowy została przedstawiona na rysunku 5.26.

Odpowiedź układu z regulatorem PID na skok jednostkowy została przedstawiona na poniższym rysunku.



Rys. 5.17 Odpowiedź układu na skok jednostkowy z regulatorem PID, nastawy otrzymane za pomocą AG

Przy użyciu AG otrzymano następujące nastawy regulatora:

Kp=9.6096 Ki=9.0518 Kd=9.6013

Oraz następujące kryteria całkowe:

IAE=1.9777 ISE=5.7456 ITAE=5.6823

5.2.8 Warunkowo stabilny system

Rozważmy następującą transmitancję:

$$K(s) = \frac{(s+6)^2}{s(s+1)^2(s+36)}$$

Powyższy obiekt jest warunkowo stabilny, zgodnie z testem Astroma[72]

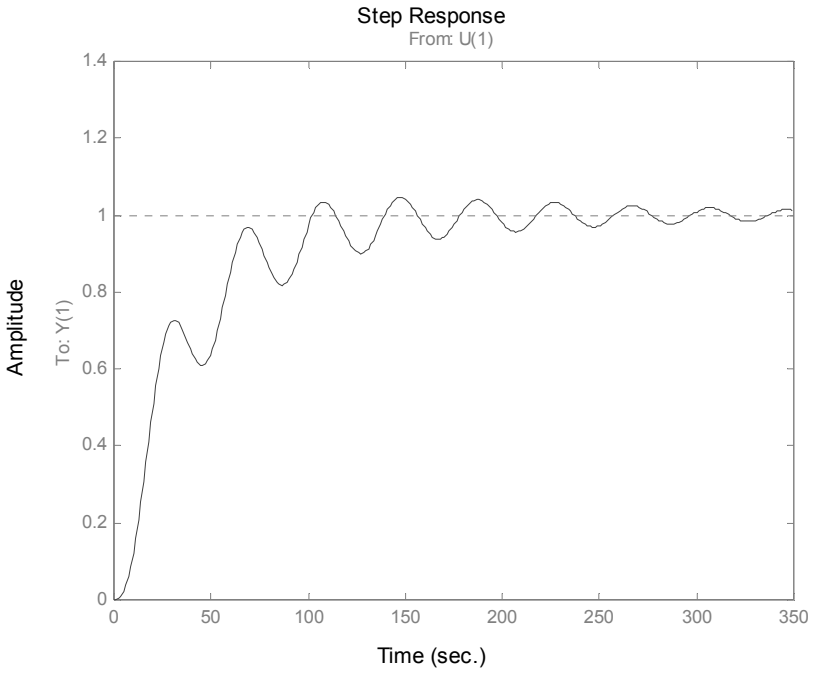
Nastawy regulatora PI otrzymane za pomocą AG, jako funkcję celu przyjęto kryterium IAE

Kp=0.07126 Ki=0.028229

Oraz kryteria błędów

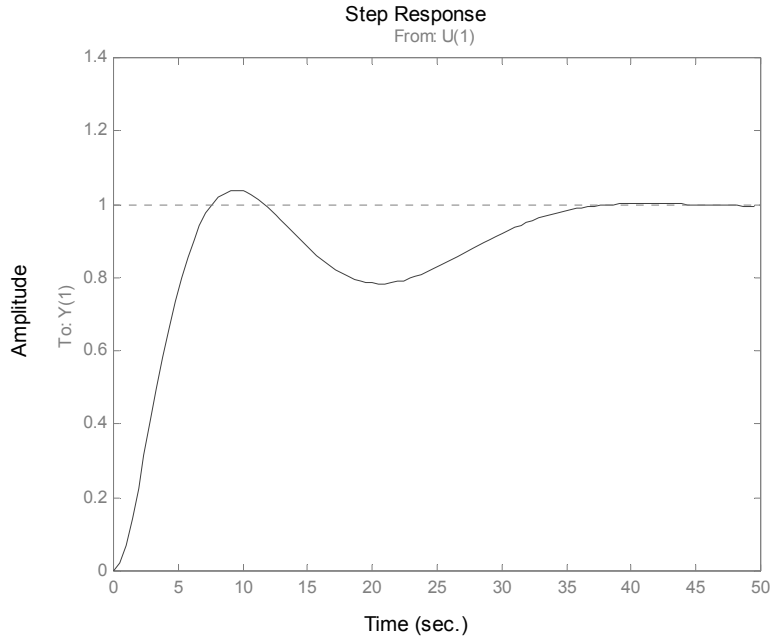
IAE: 32.0268 ISE:15.9061 ITAE:1732

[72] K. J. Astrom, oraz więcej. Hagglund (1995), *PID Controllers, theory, design and tuning,* Instrument Society of America, Research Triangle Park, North Carolina.



Rys. 5.18 Odpowiedź układu na skok jednostkowy z regulatorem PI, nastawy otrzymane za pomocą AG

Jak widać na powyższym wykresie regulator PI nie daje satysfakcjonującego rozwiązania. W tym przypadku wydaje się konieczne zastosowanie regulatora PID



Rys. 5.19 Odpowiedź układu na skok jednostkowy z regulatorem PID, nastawy otrzymane za pomocą AG

Dla regulatora PID otrzymano następujące nastawy:

Kp=1.3136 Ki=0.1552 Kd=9.7226

Oraz kryteria całkowe:

IAE: 14.3667 ISE: 6.6844 ITAE: 159.4455

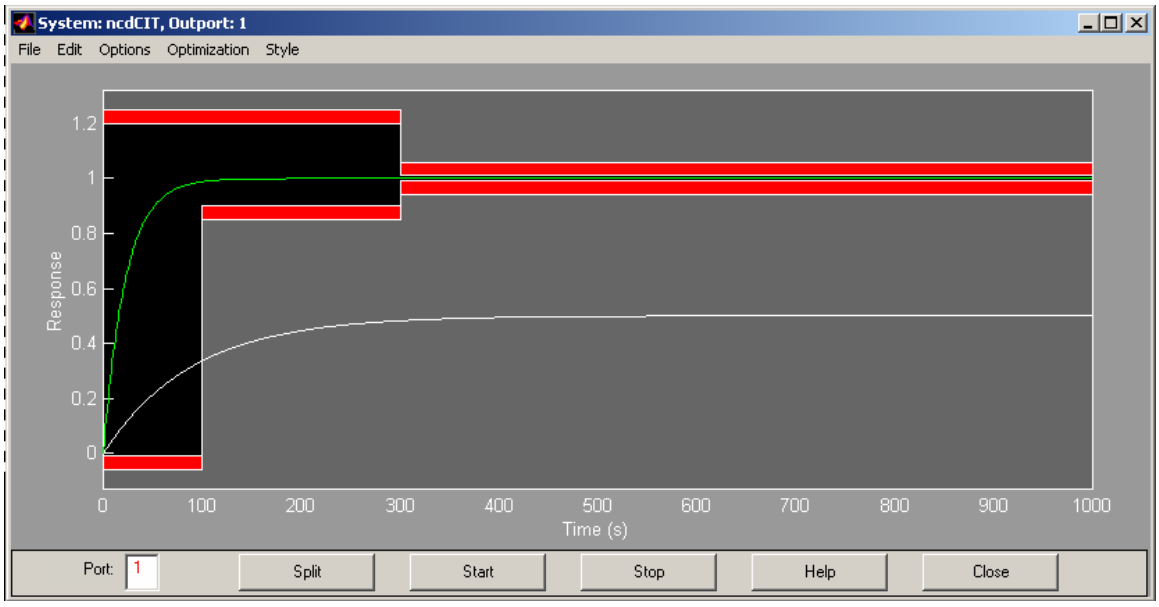
5.3 Strojenie regulatora przy wykorzystaniu NCD

W tym podrozdziale przeanalizowano strojenie regulatora dla typowych obiektów automatyki przy wykorzystaniu tego oprogramowania. Opis toolboxu został zamieszczony w rozdziale 4.2.[73]

5.3.1 Obiekt pierwszego rzędu

Poniżej rozważono strojenie regulatora dla następującej transformaty $K(s) = \frac{1}{180s+1}$

Transformata ta opisuje obiekt inercyjny pierwszego rzędu, na przykład zbiornik. Na poniższym rysunku kolorem białym zaznaczono funkcję, która podlegała optymalizacji, natomiast kolorem zielonym – odpowiedź układu na skok jednostkowy z nastrojonym regulatorem. W obu przypadkach charakterystyki są przedstawione w układzie zamkniętym.



Rys. 5.20 NCD blockset z długim czasem ustalania

Czas obliczeń 73.76 [s]

Współczynniki wzmocnienia regulatora PID:

Kp=8.0221 Ki=0.0446 Kd=-0.2544

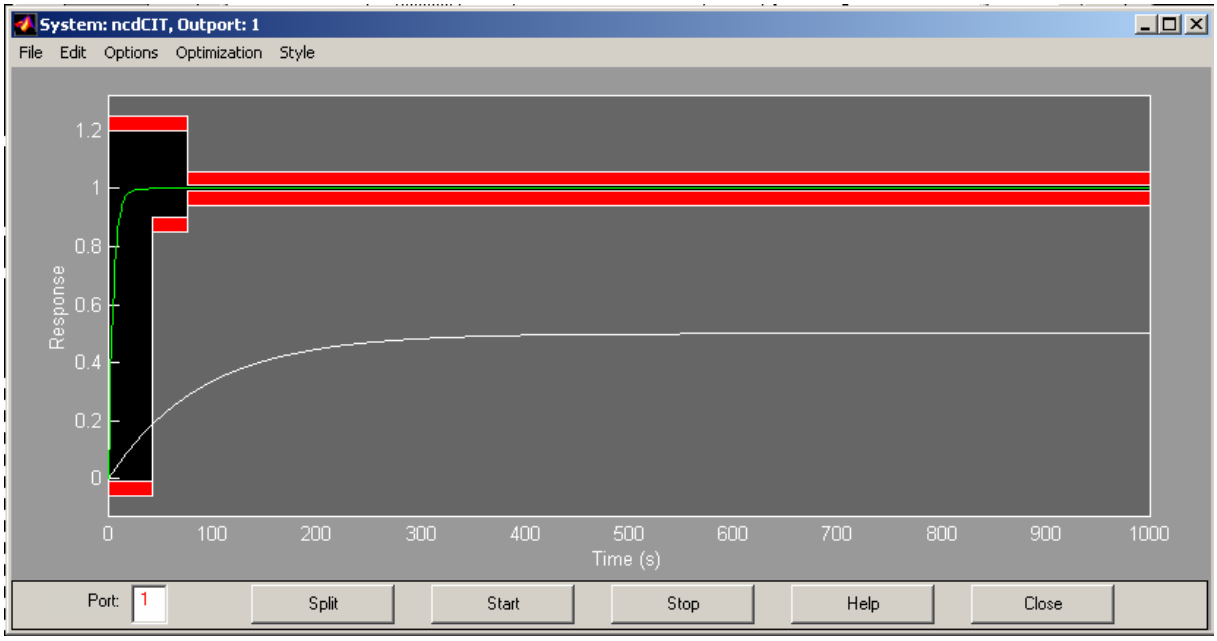
[73] więcej na ten NCD w podręczniku użytkownika Matlaba *Nonlinear Control Design Blockset* User's Guide

Kryteria całkowe:

IAE= 22.4054 ISE= 11.2029 ITAE = 502.0037[74]

5.3.2 Obiekt pierwszego rzędu z krótkim czasem ustalania

W tym przypadku zawężono zakresy tak, aby zmniejszyć czas ustalania. NCD znalazło rozwiązanie z większym współczynnikiem wzmocnienia Kp.



Rys. 5.21 NCD blockset z krótkim czasem ustalania

Współczynniki wzmocnienia regulatora PID:

Kp=38.81 Ki=0.2164 Kd=-0.59981

Kryteria całkowe:

IAE= 4.1627 ISE= 1.8586 ITAE= 24.1821

Z porównania tego przykładu z przykładem zamieszczonym w poprzednim rozdziale wynika, że kryteria całkowe są mniejsze, jednakże współczynnik wzmocnienia Kp jest większy. Możliwość kształtowania charakterystyki w sposób przyjazny dla użytkownika jest niewątpliwą zaletą NCD blockset.

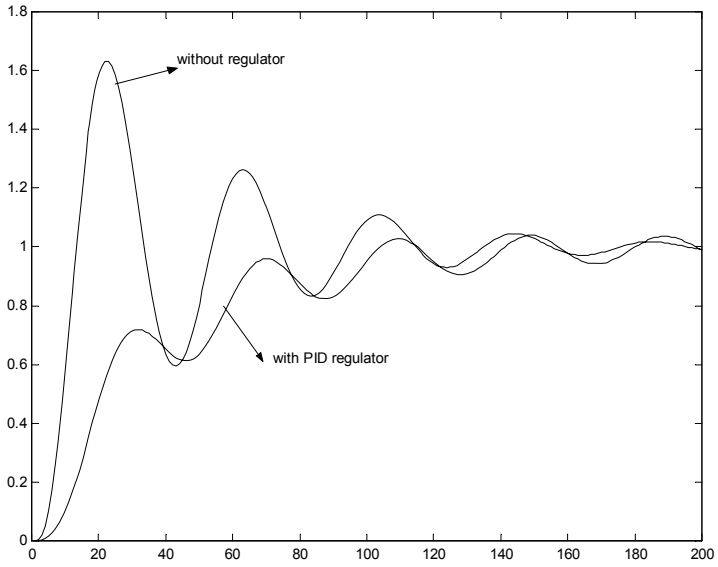
[74] czas symulacji w Simulinku wynosi1000 s.

5.3.3 Obiekt oscylacyjny

W niniejszym rozdziale rozważono strojenie regulatora dla obiektu opisanego transformatą:

$$K(s) = \frac{1}{50s^3 + 43s^2 + 3s + 1}$$

Strojenia za pomocą algorytmu genetycznego dla powyższej funkcji przejścia przeanalizowano w rozdziale 5.1.5



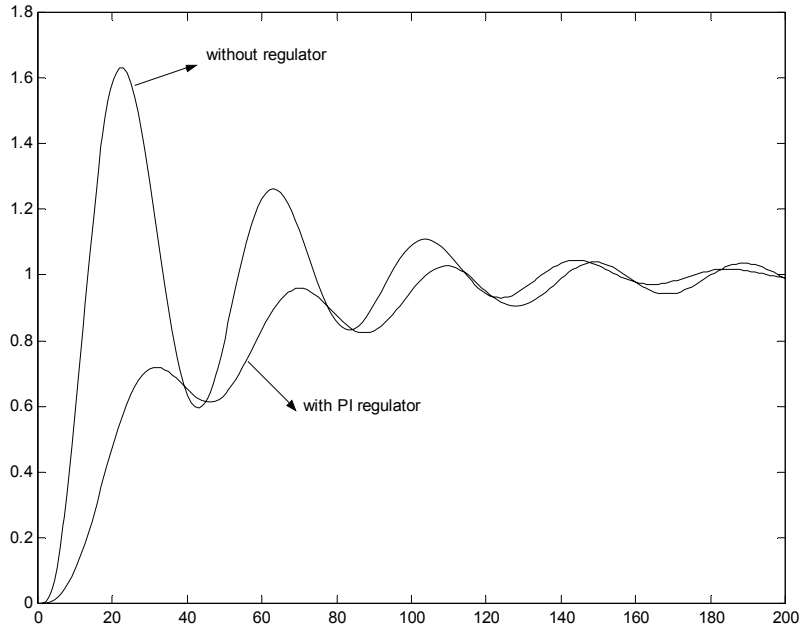
Rys. 5.22 Odpowiedź układu na skok jednostkowy, współczynniki regulatora PID otrzymane za pomocą NCD

Współczynniki wzmocnienia regulatora PID otrzymane za pomocą NCD:

Kp=0.694 Ki=0.0573 Kd=2.26

Kryteria całkowe:

IAE=40.741 ISE=19.6519 ITAE=2565



Rys. 5.23 Odpowiedź układu na skok jednostkowy,
współczynniki regulatora PI otrzymane za pomocą NCD

Współczynniki wzmocnienia regulatora PI:

Kp =0.26214 Ki=0.027916

Kryteria całkowe:

IAE= 40.7471 ISE=19.6519 ITAE=2565

NCD Blockset może wspomagać projektowanie oraz identyfikowanie skomplikowanych systemów, uwzględniając wyznaczanie wzmocnienia oraz rozwiązywanie wielu problemów optymalizacyjnych.

Programowanie sekwencyjne (*ang. Sequential Quadratic Programming (SQP)*) opiera się na metodzie *state–of–the–art*, jako nieliniowej metodzie programowania, pozwala ona korzystać z metody Niutona dla rozważanych problemów optymalizacyjnych. W każdej głównej iteracji przybliżenie jest osiągane za pomocą modyfikacji Hessiana funkcji Lagrangiana przy użyciu metody Niutona. Jest to osiągnięte za pomocą programowania QP. Główną ideą tej metody optymalizacyjnej jest sformułowanie QP opierając się na przybliżeniu funkcji Lagrangiana.

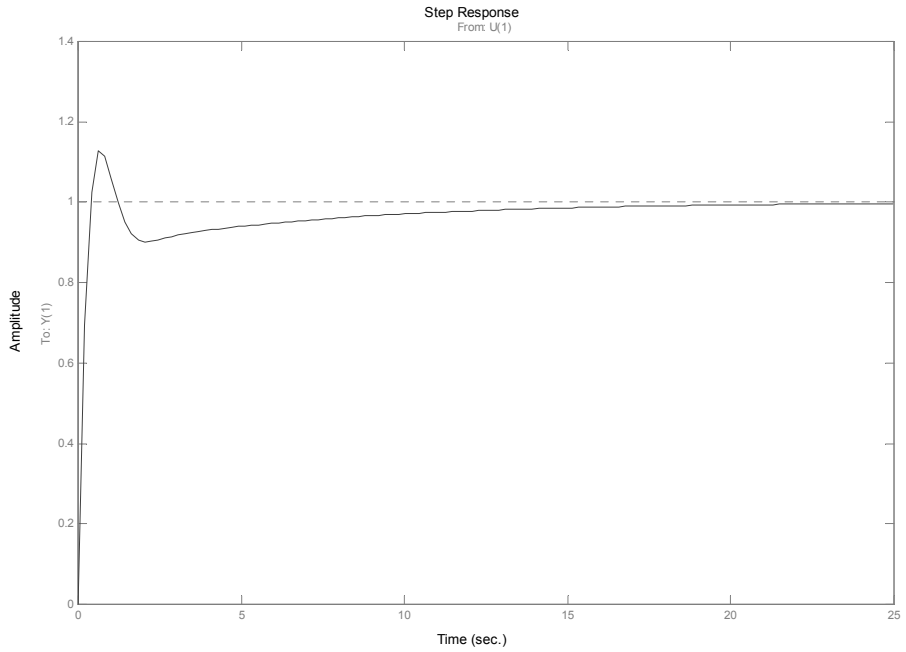
$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_1 * g_1(x)$$

5.3.4 Obiekt na granicy stabilności

Rozważmy następującą transmitancję:

$$K(s) = \frac{1}{s^2 + 1}$$

Odpowiedź układu na skok jednostkowy została przedstawiona na rys. 5.26



Rys. 5.24 Odpowiedź układu na skok jednostkowy z regulatorem PID, nastawy otrzymane za pomocą NCD

Przy użyciu NCD otrzymano następujące nastawy regulatora:

Kp=8.4397 Ki=1.2867 Kd=4.5154

Oraz następujące kryteria całkowe:

IAE=4.9192 ISE=1.2972 ITAE=24.93

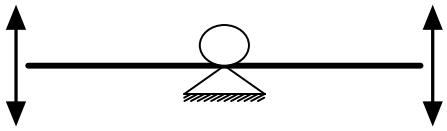
5.4 Metoda Zieglera Nicholsa

5.4.1 Obiekt na granicy stabilności

Rozważmy następującą transmitancję:

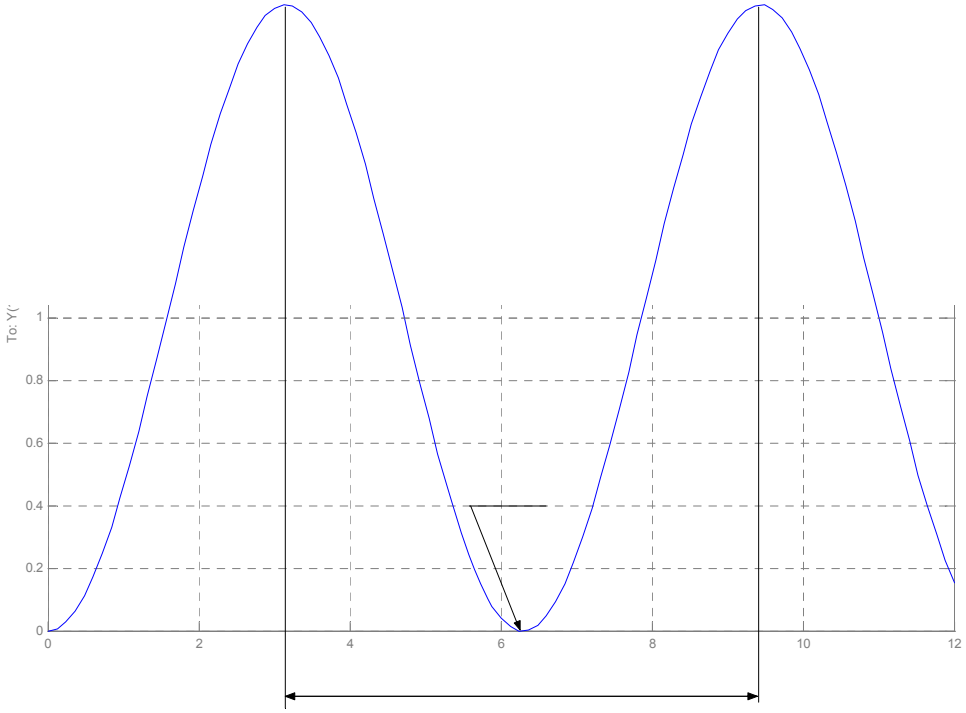
$$K(s) = \frac{1}{s^2 + 1}$$

Jest ona przykładem kuli umieszczonej na równoważni



Rys. 5.25 Kula na równoważni

Odpowiedź układu na skok jednostkowy została zamieszczona poniżej.



Rys. 5.26 Odpowiedź układu na skok jednostkowy dla kuli umieszczonej na równoważni

Tosc=9.43-3.12=6.31

k_{kr}=6.3

Nastawy regulatora PID:

Kp=0.45*k_{kr}=0.45*6.3=2.835

Ti=0.83*T_{osc}=0.88*6.31=5.237 Ki=1/Ti=0.19094

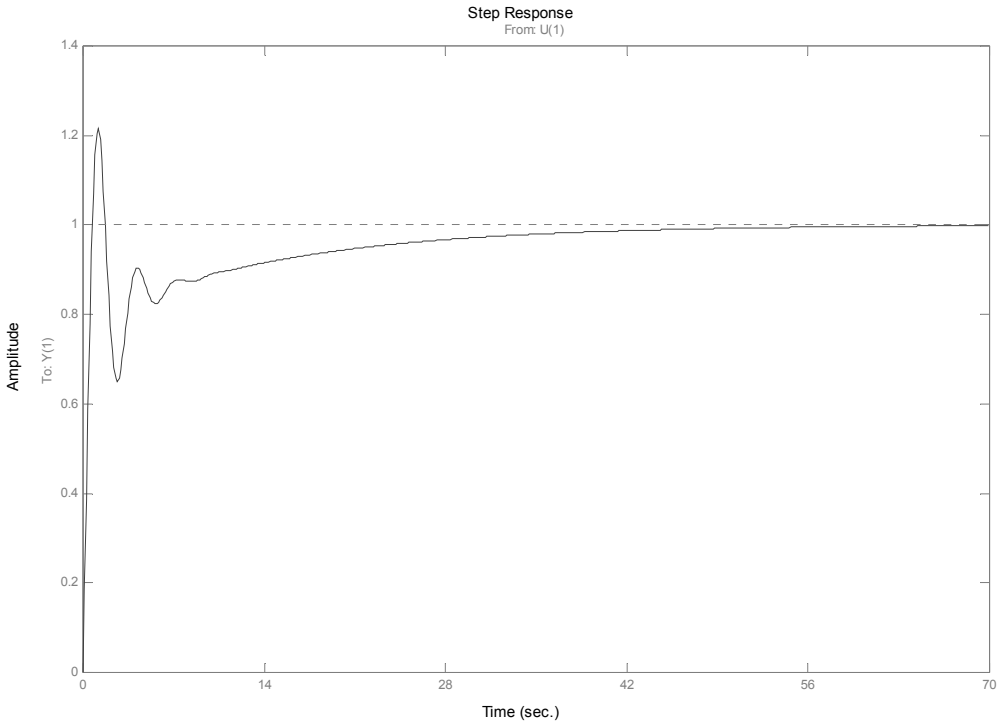
Obiekt w układzie zamkniętym jest niestabilny z regulatorem PI, w tym przypadku konieczne jest zastosowanie regulatora PID.

Nastawy regulatora PID:

$Kp=0.6*k_{kr}=0.6*6.3=3.78$

$Ti=0.5*T_{osc}=0.5*6.31=3.155$ $Ki=1/Ti=0.31696$

$Td=0.125*T_{osc}=0.125*6.31=.78875$ $Kd=1/Ti=1.2678289$



Rys. 5.27 Odpowiedź układu na skok jednostkowy, nastawy regulatora PID otrzymane za pomocą metody Zieglera-Nicholsa

5.5 Porównanie metod strojenia regulatorów

Obecnie w automatyce występuje 181[75] metod strojenia regulatorów PID. Niemożliwe jest przeanalizowanie wszystkich metod strojenia regulatorów w ramach tej pracy magisterskiej. W Optimization toolbox dostępne jest sześć metod poszukiwania optimum funkcji.[76] W niniejszej pracy przeanalizowano dwie metody strojenia regulatorów, tj. algorytm genetyczny

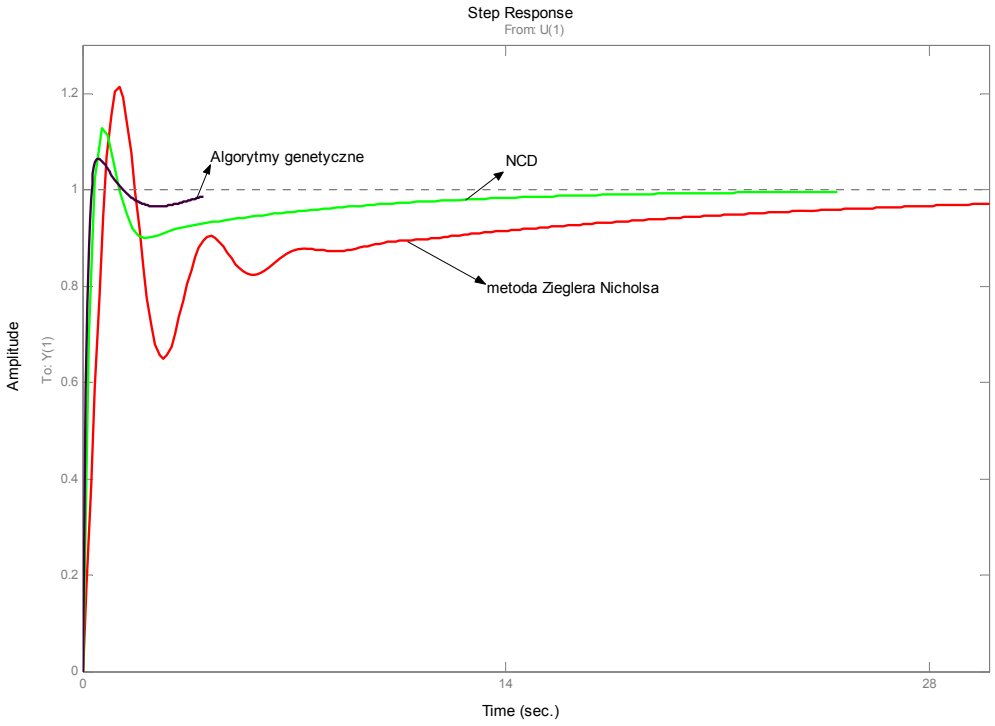
[75] Dane na podstawie korespondencji z Aidanem O'Dwyerem, *Dublin Institute of Technology*. aidan.odwyer@dit.ie

[76] Polecenie *bandem* przywołuje demo Optimization Toolbox, wykorzystane do zoptymalizowania funkcji Rosenbrock (*ang. banana function*).

oraz NCD. Zawarto również podstawowe informacje na temat tradycyjnej metody Zieglera - Nicholsa. Nie można jednoznacznie powiedzieć, która z metod strojenia regulatora jest najlepsza. Wszystko zależy od rodzaju przyjętych kryteriów. Do zalet algorytmów genetycznych należy fakt, iż znajdują one bardzo szybko globalne minimum. Jeżeli zależy nam na strojeniu regulatora w czasie rzeczywistym (czas obliczeń musi być bardzo krótki), to można zastosować algorytmy genetyczne do znalezienia globalnego minimum, natomiast dostrojenie regulatora za pomocą metody najmniejszych kwadratów. Generalnie AG stosuje się do skomplikowanych problemów optymalizacyjnych, tam gdzie występuje wiele zmiennych, np. problem komiwojażera.

Do zalet NCD należy przede wszystkim fakt, iż zaimplementowanie tej metody do optymalizacji jest stosunkowo łatwe. Niestety toolbox NCD nie jest oprogramowaniem darmowym, w przeciwieństwie do toolboxu AG[77]. Obecnie toolbox AG nie jest oficjalnym produktem Mathworks.

Czas obliczeń przy wykorzystaniu NCD jest dłuższy niż dla AG.



Rys. 5.28 Porównanie metod strojenia regulatorów

Powyższy rysunek przedstawia porównanie metod strojenia regulatorów dla obiektu oscylacyjnego, przedstawionego transformatą:

[77] ftp://ftp.mathworks.com/pub/tech-support/solutions/s1248/genetic/

$$K(s) = \frac{1}{s^2 + 1}$$

Nastawy regulatora otrzymane poszczególnymi metodami zostały zamieszczone w poprzednich rozdziałach.

5.6 Graficzny Interfejs Użytkownika

Jednym z celów pracy jest zaprojektowanie interfejsu graficznego (*ang. Graphic User Interface, GUI*). Interfejs ten powoduje, iż obsługa programu jest bardziej przyjazna dla użytkownika. Nie jest on zobligowany do zrozumienia kodu źródłowego programu, a tym samym przyswojenia sobie wielu zmiennych. Interfejs ten powoduje, iż program może być obsługiwany przez wielu użytkowników, a nie tylko przez autora programu. Jest to niewątpliwą zaletą. Oczywiście są również wady. Jedną z nich jest fakt, że program znajduje zastosowanie w zawężonym zakresie. Dla przykładu identyfikacja jest tylko dla obiektów pierwszego rzędu. Rozszerzenie identyfikacji o obiekty wyższych rzędów zwiększyłoby uniwersalność programu, jednakże użytkownik byłby zmuszony do zdefiniowania większej ilości zmiennych (*np. theta format*), czyli obsługa programu byłaby bardziej skomplikowana.

GUI składa się z czterech podstawowych pól:

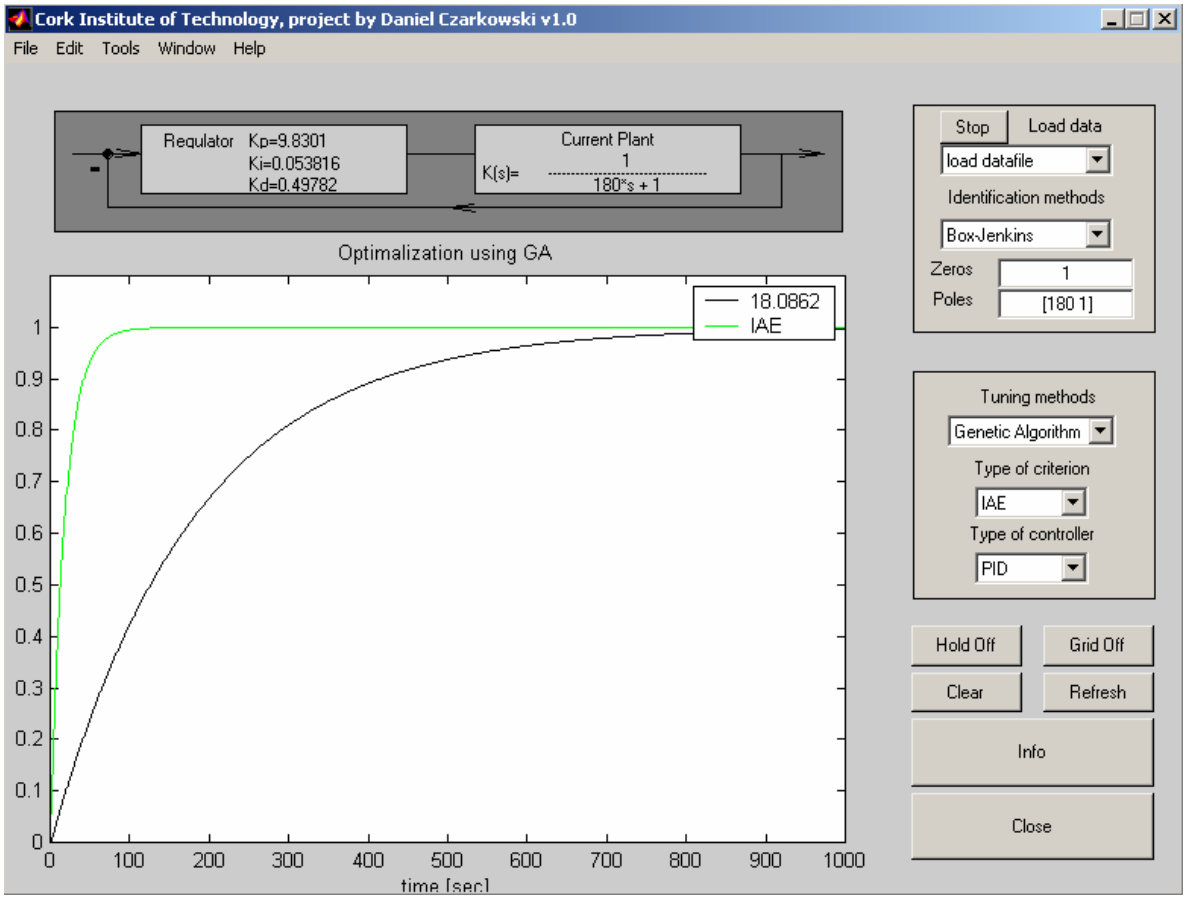
1) Identyfikacja

2) Strojenia regulatora

3) Pętla sprzężenia zwrotnego

4) Wykres odpowiedzi układu na skok jednostkowy

Na poniższym rysunku jest przedstawione okno główne programu. Po prawej stronie tego okna znajdują się dwa pola. Pierwsze z nich jest polem, w którym identyfikujemy obiekt.



Rys. 5.29 Graficzny Interfejs Użytkownika

1. Identyfikacja[78] W polu tym są dostępne trzy menu. W pierwszym z nich mamy do wyboru dwie opcje: *load data* oraz *Real time: tank.* W przypadku wybrania *load data*, program odwołuje się do pliku `datafile.mat`, w którym są zapisane dane otrzymane z rzeczywistego obiektu. Natomiast wybór *Real time: tank*, umożliwia identyfikację obiektu w czasie rzeczywistym. Aby to osiągnąć, stacja robocza musi być wyposażona w *Data Acquisition Unit* oraz *Real Time Toolbox ver. 3.0* dostarczone przez czeską firmę HUMUSOFT[79]. Kod programu identyfikującego obiekt w czasie rzeczywistym został zamieszczony w załącznikach pod nazwą `ident_rt`. Przycisk *stop* umożliwia przerwanie procesu identyfikacji w czasie rzeczywistym. W polu identyfikacji dostępna jest również opcja wprowadzenia zer i biegunów przez użytkownika. Umożliwia to przeprowadzenie strojenia regulatora dla pożądanego obiektu, wprowadzonego przez użytkownika bądź zidentyfikowanego na podstawie danych umieszczonych w pliku `datafile.mat`. Wprowadzone zera i bieguny są wyświetlane w pętli sprzężenia zwrotnego. Również dane obiektu otrzymane w wyniku procesu

[78] Szerzej w rozdziale 2.1

[79] oficjalna strona producenta http://www.humusoft.com/

identyfikacji zostają uwidocznione w tym polu. W menu *metody identyfikacji* dostępne są trzy metody identyfikacji:

- klasyczna,
- Box – Jenkins,
- najmniejszych kwadratów.

Metody te zostały omówione w rozdziale 5.1

2. Strojenie regulatora[80] W tym polu dostępne są trzy różne menu. Pierwsze z nich, metody strojenia regulatora (*ang. Tuning methods*), umożliwia dostęp do następujących metod:

- Algorytmy genetyczne,
- Nonlinear Control Design Blockset, NCD.

Po wybraniu opcji strojenia regulatora za pomocą algorytmów genetycznych program przywołuje plik `genhaploid.m` i w oknie Matlaba można zaobserwować kolejne iteracje programu. Wyniki obliczeń (współczynniki Kp Ki Kd) są umieszczone w oknie sprzężenia zwrotnego. Natomiast opcja NCD otwiera okno w Simulinku. Metoda strojenia regulatora opiera się na *state–of–the–art optimization*[81]. W celu poprawnego działania programu, Matlab musi posiadać *Optimization toolbox* oraz *Nonlinear Control Design Blockset*. Po wybraniu rodzaju regulatora oraz kryterium doboru regulatora należy wybrać metodę optymalizacji. Sposób strojenia został opisany w rozdziale 4.3.

W następnym menu, funkcja przystosowania algorytmów genetycznych wg kryteriów (*ang. type of criterion*), dostępne są opcje:

- IAE
- ISE
- ITAE

Są to kryteria według których algorytmy genetyczne obliczają funkcję kosztów.[82] Kolejnym menu w polu strojenia regulatora jest rodzaj regulatora. Do wyboru mamy dwie opcje:

- PID
- PI

Rodzaje regulatorów oraz podstawowe informacje na ten temat zostały zamieszczone w rozdziale trzecim.

---

[80] Szerzej w rozdziale 4

[81] Szerzej w *Matlab Optimization Toolbox User's Guide*

[82] Szerzej w rozdziale 3.4

3. W pętli sprzężenia zwrotnego widać funkcję przejścia (*ang. transfer function*) opisującą model rzeczywisty obiektu optymalizowanego oraz współczynniki regulatora PID.

4. Ostatnią opcją jest wykres odpowiedzi układu na skok jednostkowy.

W programie umieszczono dodatkowe przyciski ułatwiające porównywanie obliczeń dla różnych metod identyfikacji bądź optymalizacji. Do przycisków tych należą:

- *Hold on / Hold off* – funkcja umożliwia umiejscowienie wielu charakterystyk na jednym wykresie,
- *Grid on / Grid off* – włącza lub wyłącza linie pomocnicze na wykresie,
- *Clear* – funkcja umożliwia wymazanie wszystkich charakterystyk zamieszczonych na wykresie,
- *Refresh* – uaktualnienie danych,
- *Help* – pomoc (dostępna w języku angielskim).

5.7 Instrukcja laboratoryjna

Przed przystąpieniem do ćwiczenia należy się zapoznać z interfejsem graficznym programu, który został opisany w poprzednim rozdziale. W celu uruchomienia programu należy wpisać w wierszu poleceń Matlaba komendę `InitialWindow` Jeżeli program nie zostanie uruchomiony należy upewnić się, że ścieżka dostępu została dodana w środowisku Matlab.[83] Po uruchomieniu programu w pierwszej części ćwiczenia należy zidentyfikować obiekt. Można to uczynić na dwa sposoby:

- Dla komputerów wyposażonych w Data Acqusition Unit oraz z odpowiednim oprogramowania należy użyć funkcji *Real time: tank*, dostępnej w polu identyfikacji.
- Dla komputerów nie posiadających odpowiedniego wyposażenia oraz oprogramowania należy załadować dane z pliku *datafile.mat.*

Po wybraniu opcji *load datafile* powinniśmy otrzymać wykres przedstawiony na rysunku 5.2, następnie należy zidentyfikować obiekt za pomocą:

- Metody klasycznej.
- Box – Jenkins.

[83] polecenie dodające ścieżkę dostępu `addpath`; można również kliknąc na *file* a następnie *set path* w oknie poleceń Matlaba

- Metody najmniejszych kwadratów.

Można również wprowadzić pożądaną transformatę za pomocą zer i biegunów.

Należy dokonać porównania otrzymanych wyników za pomocą dostępnej funkcji HOLD ON.

Następnym krokiem w ćwiczeniu jest zapoznanie się z metodami poszukiwania nastaw regulatora PID. Do przeanalizowania są dwie metody: Algorytmy genetyczne oraz NCD.

Po zidentyfikowaniu obiektu wybieramy kryteria według których algorytm będzie optymalizował. Następnie wybieramy z menu *tuning methods* opcję *Genetic Algorithm*. W przestrzeni roboczej Matlaba możemy obserwować kolejne iteracje algorytmu.

Kolejną metodą optymalizacyjną jest NCD. Metoda ta oraz sposób korzystania z niej został opisany w rozdziale 4.2.

W ćwiczeniu należy znaleźć nastawy regulatora PI(D) używając dwóch metod strojenia regulatora dla następujących obiektów.

$$K(s) = \frac{1}{180s + 1}$$

$$K(s) = \frac{1}{50s^3 + 43s^2 + 3s + 1}$$

oraz $K(s) = \frac{1}{s^2 + 1}$

Zakończenie

Wszystkie zadania ujęte w zakresie pracy zostały zrealizowane, jednakże praca nie wyczerpała wszelkich zagadnień związanych z identyfikacją oraz strojeniem parametrów regulatora PID. Rezultatem tej pracy jest program, który identyfikuje obiekt i poszukuje nastaw regulatora PID. Pierwszy krok stanowiło przesłanie danych otrzymanych z odpowiedzi układu na skok jednostkowy. Dane te były zakłócone, a zatem konieczne okazało się zastosowanie filtracji. W niniejszej pracy dużą część czasu poświęcono na komunikację pomiędzy Matlabem a obiektem rzeczywistym. Po przesłaniu danych do komputera zidentyfikowano obiekt trzema metodami oraz porównano uzyskane funkcje przejścia z rzeczywistymi danymi. Różnice były niewielkie, wynosiły około 1%, jednakże najlepszą okazała się metoda Box – Jenkins, dostarczona wraz z *Identification Toolbox*. Kolejnym krokiem było strojenie parametrów regulatora PI(D) za pomocą algorytmów genetycznych oraz porównanie tej metody z NCD. Nie można jednoznacznie stwierdzić, która z metod strojenia regulatora PID jest bardziej niezawodną. Na pewno więcej pracy wymagało zaimplementowanie algorytmów genetycznych do optymalizacji niż używanie oprogramowania stworzonego do rozwiązywania problemów optymalizacyjnych, jakim jest NCD.

Ze względu na dydaktyczny charakter pracy stworzono Graficzny Interfejs Użytkownika. Ułatwia on korzystanie z programu, ponieważ nie ma potrzeby definiowania zmiennych w przestrzeni roboczej Matlaba. Dodatkowo wszystkie operacje mogą być powtarzane wielokrotnie.

Integralną i fundamentalną częścią pracy są pliki załączone do dokumentu. To one stanowią kwintesencję tej pracy magisterskiej, która powstała w ramach stypendium Erasmusa realizowanego podczas semestralnych studiów w *Cork Institute of Technology* w Irlandii.

W przyszłości można moją pracę rozbudować o test Astroma, czyli zaimplementować jedenaście podstawowych obiektów występujących w przemyśle do programu, następnie wykorzystać algorytmy genetyczne bądź inne metody dostępne w *Optimisation Toolbox*, do poszukiwania nastaw regulatora PID.

Literatura

1. Astrom K.J., Hagglund, *The Future of PID control,* Control Engineering Practice 9 (2001) pages 1163-1175
2. Astrom, K. J., Eykoff, P., *System Identification – a survey*, Automatica, 7, 1971, pages 123–167
3. Brogan W.L., *Modern Control Theory*, Third edition, Prentice Hall
4. Camacho E.F. and Bordons C., *Model Predictive control in the process industry*, Prentice Hall
5. Chotkowski W., *Podstawy automatyki,* Skrypt Politechniki Gdańskiej
6. *Control Engineering Practice*, Volume 9, Issue 11 November 2001 pages 1159–1161
7. Czarkowski D., *Identification and Optimization PID parameters using MATLAB,* DLX4 Project, Cork Institute of Technology, 2002
8. Deasy P., *PID Parameter optimisation using Genetic Algorithms – The Astrom Benchmark Test*, DLX4 Project 2001 CIT
9. Delores M. Etter, *Engineering Problem Solving with Matlab*, second edition, Prentice Hall, New Jersey 1997
10. Fatla K., *Optymalizacja nastaw regulatora wzbudzenia generatora synchronicznego, przy pomocy algorytmów genetycznych*, Praca magisterska, Politechnika Wrocławska 1999/2000
11. Fatla K., *Using Genetic Algorithms for Optimaization of PID Controller,* DLX4 Project 1999 CIT
12. Findeisen W., *Technika regulacji automatycznej*, PWN Warszawa, 1978r.
13. Gauss K. F., *Teoria Motus Corporum Coelestium in Sectionibus Conicus Solem Ambientum* 1809 (tłumaczenie: Theory of motion of the heavenly bodies moving about the sun in conic section, Dover, New York)
14. http://galaxy.uci.agh.edu.pl/~ttward/iden/ Laboratory of identification object 13/03/02
15. http://www.humusoft.com/
16. http://www.mathworks.com
17. http://www.sciencedirect.com
18. Man K.F. Tang K.S. Kwong S., *Genetic Algorithms – concepts and designs,* Spinger 1999
19. *Mathematics and Computers in Simulation* Volume 51, Issues 3–4 January 2000 pages 287–300
20. *Matlab Build Graphic User Interface GUI,* User's Guide

21. Matlab Control System Toolbox, User's Guide

22. Matlab Nonlinear Control Design Blockset, User's Guide

23. Matlab Optimization Toolbox, User's Guide

24. Matlab System Identification Toolbox, User's Guide

25. Mościcki J., Ogonowski Z., *Advance Control with Matlab & Simulink* Ellis Horwood 1995

26. Mulawka J., *Systemy Ekspertowe,* WNT, Warszawa 1996r.

27. Munoz I. M., *System Identification Using Genetic Algorithm* DLX4 Project 2001 CIT

28. O'Dwyer A., Ph.D. thesis, *The estimation and compensation of process with time delays*, Dublin City University, 1996

29. O'Mahony T., Downing C. J., Fatla K., *Genetic Algorithms for PID Parameter Optimisation: Minimising Error Criteria*

30. O'Mahony T., Downing C. J., *Real–time fluid level control using PID and GPC,* Proc. First International Postgraduate Research Students Conf. DIT Dublin 1998

31. O'Mahony T., Downing C. J., *Simulink for PID controller design – goldmine or minefield?* Proc. Irish Signals & Systems Conf. 2000, UCD, Dublin p. 450 – 457

32. O'Mahony T., *Real–time Control of a Single–tank System using Humusoft* Electronics Dept., CIT 2000

33. Ogata K., *Modern Control Engineering*, fourth edition, Prentice Hall, New Jersey 2002

34. Perycz S., *Podstawy automatyki,* Skrypt dla instytutu okrętowego Politechniki Gdańskiej, Gdańsk 1980r.

35. Rutkowska D., Piliński M., Rutkowski L., *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte,* PWN, Warszawa 1999r.

36. Shinners S., *Modern control system theory and design*, second edition, A Wiley Interscience Publication,

37. Söderström T., Stoica P., *System Identification*, Prentice Hall International 1994

38. Tecquipment Limited http://www.tq.com/index.html

39. Węgrzyn S., *Podstawy automatyki,* PWN, Warszawa 1974r.

40. Żelazny M., *Podstawy automatyki,* PWN, Warszawa 1976r.

41. Ziegler, J.G. and Nichols, N.B. *Optimum settings for automatic controllers circuit*. Transaction of the ASME*,* 1942 November, pages 759–768

Załączniki

Załącznik A m–pliki

ident_rt.m

```
% ident_rt.m
%
% measure data and filtering from a plant in real time
%
% Author:      Daniel Czarkowski DLX4
%                            daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department


%Variables
SamplingPeriod=1;            % sampling period
N=1000;                      % number of samples
u=1*zeros(N,1);              % prepare the output vector
Vout=zeros(N,1);

rtclear;                                          % clear drivers
rtload('ad512',256,[0 3 3 3 3 3 3 3]);  % load the HUMUSOFT AD512 driver
rtdef(1,'out',SamplingPeriod,1);     % define output timer
rtdef(2,'in',SamplingPeriod,1);      % define input timer
rtstart all;                         % start the timers


%Main loop
tic;                         % Start a stopwatch timer
disp 'busy'
for i=1:N;
   if round(rttool('Read',1,'Run'))==1
        while (rtrd(2,'t'))<i; end;   % wait for the next sample
               rtwr(1,'y',u(i));
               Vout(i)=rtrd(2,'y');  % read the input value
               Voutend=round(Vout*1000000);  % round and multimle
      if i>120
         and_=and(Voutend(i-1)==Voutend(i),Voutend(i-120)==Voutend(i));
         if and_==1   % stop pump when two samples are equal
            disp 'OK'
            rtstop (1)
                Na=i;               % measure number of samples
         end;
      end;
   end;
end;
MeasurementTime=toc   % stop a stopwatch timer
```

```
rtwr (1,-1);                                % stop a pump
rtstart (1)                                         % start timer 1
try
   Na;
catch
   Na=N;
end

Vout=Vout(1:Na);                                            % initial matrix
N=Na                                                        % number of samples

Vout1=(Vout(1)+Vout(2))/2;                          % mean value saples 1 & 2
VoutOffset=Vout-Vout1;                              % subtract offset
Vout=VoutOffset;                                            % new data without offset

%http://www.mathworks.com/access/helpdesk/help/techdoc/ref/polyfit.shtml
%Polynomial curve fitting
x=[1:SamplingPeriod:N]';
p = polyfit(x,Vout,5);                                      % fit polynomial to data
ypoly = polyval(p,x);                                       % measured data after estimation
%table = [x y f y-f];
%y = erf(x);                                        % error function
clear p Na Voutend SamplingPeriod i and_ u Vout1
disp 'Done measure process'
plot (x,Vout,'r',x,ypoly,'b');
axis auto;

disp ('error of approximation[%]')
sum((abs(Vout-ypoly)./Vout))/length(ypoly)*100
title('Measured (red), Polynomial curve fitting (blue)');
save datafile;                % spare datafile is called 1000.mat
```

rtstop1.m

```
% rtstop1.m
%
% emergency stop of measuring data
%
% Author:      Daniel Czarkowski DLX4
%                              daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department

clear all
clc
rtclear;
Ts = 0.01;                          % sampling period
rtload('ad512',256,[0 3 3 3 3 3 3 3]);% load the HUMUSOFT AD512 driver
rtdef(1,'out',Ts,1);                                % define output timer
rtdef(2,'in',Ts,1);                                 % define input timer
```

```
rtwr(1,-1);                      % write the initial condition
rtstart all                          % start timers
```

ident_calc.m

```
% ident_calc.m
%
% identification of a plant from datafile
%
% Author:      Daniel Czarkowski DLX4
%                             daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department


%Variables
u=ones(N,1);                         % u-input  ypoly-output
z=[ypoly u];                         % data from file datafile.mat
                                            % or script ident_rt.m

if IdentMethods==1           %identification using classical method
k=(ypoly(N)-ypoly(1))/1; %(u(N)-u(1));
stepchange=(1-exp(-1))*k;     % 63% of settling value
final_value=(stepchange+ypoly(1));
for i=1:N;                           %find the first sample
   if ypoly(i) <= final_value;
      ts=x(i);
   end
end
disp 'plant computes using a classical method'
PlantS=tf([k],[ts 1]) % transfer function of a plant
sim('pid_fine_ol');          % Simulate a simulink model
plot(yout,'r');
title('Classical method (red)');


elseif IdentMethods==2
%Computes the prediction error estimate of a Box-Jenkins model.
th=bj(z,[1 0 0 1 0]); % z-meassured data
[Num,Den]=th2tf(th);% transforms from the THETA-format to transfer functions.
disp 'Plant computes using a Box-Jenkins model'
plantbj=tf(Num,Den,1);
PlantS=d2c(plantbj,'zoh')     % conversion of discrete LTI models to continuous time.
                                    % 'zoh' Assumes zero-order hold on the inputs.
sim('pid_fine_ol');          % Simulate a simulink model
plot(yout,'g');
title('Box-Jenkins (green)');

elseif IdentMethods==3
%Computes LS-estimates of ARX-models.
disp 'plant computes using LS'
```

```
th=arx(z,[1 1 1]);           %  Computes LS-estimates of ARX-models.
[Num,Den]=th2tf(th);
plantarx=tf(Num,Den,1);
PlantS=d2c(plantarx,'zoh')    % conversion of discrete LTI models to continuous time.
                                    % 'zoh' Assumes zero-order hold on the inputs.
sim('pid_fine_ol');          % Simulate a simulink model
plot(yout,'b');
title('Least squares (blue)');
end
clear IdentMethods plantarx plantbj th ts i z u
main;
```

ncdinit.m

```
% ncdinit.m
% ncdinit Sets up necessary data files for optimization of ncdCIT.mdl.
%
%
% Author:      Daniel Czarkowski DLX4
%                              daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department
%
% Define Optimization parameters
%     - incremental step size  : ncdStruct.Tdelta
%     - tunable parameters     : ncdStruct.TvarStr
%     - optimization options   : ncdStruct.OptmOptns
%     - lower bound limits     : ncdStruct.TvlbStr
%     - upper bound limits     : ncdStruct.TvubStr


ncdglob; % declare globals

ncdStruct.Tdelta = 1;
ncdStruct.OptmOptns = [1 0.001 0.001];
ncdStruct.TvlbStr = '';
ncdStruct.TvubStr = '';
try
   contr;
catch
   contr=1
end

if contr==1
   ncdStruct.TvarStr = 'Kp Ki Kd';
   disp('PID')
else
   ncdStruct.TvarStr = 'Kp Ki';
   disp('PI')
end
```

```
try
   PlantS
catch
PlantNum=1;
PlantDen=[180 1];
end

Kp=1; % Define Tunable Variable initial values
Ki=0;
Kd=0;

ncdCIT;                              %open simulink model
disp('Done initializing')
```

genhaploid.m

```
% genhaploid.m
%
% Genetic Algorithm implemented into PID regulator
% This script calls Genetic Algorithm Toolbox
% Download toolbox from:
% ftp://ftp.mathworks.com/pub/tech-support/solutions/s1248/genetic/
%
% Author:      Daniel Czarkowski DLX4
%                           daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Departmenttry

try
PlantNum;
catch
PlantNum=[1];                %default Numerator
PlantDen=[180 1];            %default Denumerator
end

try
   ErrorEstim;               % Type of performance index
catch                                    % 1-IAE 2-ISE 3-ITAE
   ErrorEstim=1              % default IAE
end
try
   contr;                            % Type of controller (1=PID 2=PI)
catch
   contr=1                           %default PID
end

try
   N;                                      % Number of samples
catch
   N=1000;
end
```

```
if contr==1
   disp('PID')
x=([0 1 0])';                                % initial values
vlb = [-10 -10 -10];          % lower bounds
vub = [10 10 10];                    % upper bounds
bits = [16 16 16];
Kd=x(1);
Kp=x(2);
Ki=x(3);
PIDgain=[Kd Kp Ki];
else
   disp('PI')
x=[1 0]';                             % initial values
vlb = [0 0];                  % lower bounds
vub = [10 10];                % upper bounds
bits = [16 16];
Kp=x(1);
Ki=x(2);
PIDgain=[Kp Ki];
end

PlantS=tf(PlantNum,PlantDen);
PIDden=[1 0];
PIDreg = tf(PIDgain,PIDden);         % Trasfer function of PID Controller
oltf= PlantS*PIDreg;                                % Open Loop Transfer Function

cltf=PIDreg*PlantS;                                 % Close Loop Transfer Function
cltf = feedback(cltf,1,-1);
[y t] = step(cltf);           % Step response of closed-loop system in time domain
error = 1-y;
if ErrorEstim==3
   error = error.*t;
   disp('ITAE');
end
error = abs(error);
error = sum(error);
if ErrorEstim==2
   error = error.^2;
   disp('IAE')
elseif ErrorEstim ==1
   disp('IAE')
end
fitness = error                      %error without PID regulator

%options = foptions([1 -1]);
options = foptions([1 1e-4]); %count up to 1e-4
options(13) = 0.03;
options(14) = 50;                                   %The default maximum generations
tic
%x0=[50 100 0];
warning off
```

```
[x,stats,options,bf,fg,lg] = ge-
netic('genpid',[],options,vlb,vub,bits,N,PlantS,ErrorEstim,contr);
warning on
toc
%STATS   - [max min mean std] for each generation
%           OPTIONS - options used
%           BESTF   - Fitness of indivadual X (i.e.: best fitness)
%           FGEN    - first generation population
%           LGEN    - last generation population
% The highest point found by the genetic algorithm is
bf;

if contr == 1
   Kp=x(2);
   Ki=x(3);
   Kd=x(1);
        PIDgain=[Kd Kp Ki];
else
   Kp=x(1);
        Ki=x(2);
        PIDgain=[Kp Ki];
end

PIDreg = tf(PIDgain,PIDden);         % Trasfer function of PID Controller

cltf=PIDreg*PlantS;
cltf = feedback(cltf,1,-1);
[y t]=step(cltf,1:1:1000);
plot(y,'g');
title ('Optimalization using GA')
xlabel ('time [sec]')
axis auto;

error = 1-y;

if ErrorEstim==1
   error = abs(error);
   error = sum(error);
   errorIAE=error;
   disp('IAE')
   legend(num2str(errorIAE),'IAE');
elseif ErrorEstim==2
   error = error.^2;
   error = abs(error);
   error = sum(error);
   errorISE=error;
   disp('ISE')
   legend(num2str(errorISE),'ISE');
elseif ErrorEstim==3
   error = error.*t;
   error = abs(error);
   error = sum(error);
   errorITAE = error;
```

```
   disp('ITAE')
   legend(num2str(errorITAE),'ITAE')
end
fitness = error;
clear errorIAE errorISE errorITEA cltf oltf bf t ulb vlb
main;
```

genpid.m

```
% genpid.m
%
% Genetic Algorithm implemented into PID regulator
% This function is called by genhaploid.m
%
% Author:      Daniel Czarkowski DLX4
%                            daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Departmenttry

function fitness = genpid(x,N,PlantS,ErrorEstim,contr)

if contr == 1 %values for PID regulator
   Kd=x(1);
        Kp=x(2);
   Ki=x(3);
        PIDgain=[Kd Kp Ki];
else                  %values for PI regulator
   Kp=x(1);
        Ki=x(2);
        PIDgain=[Kp Ki];
end



PIDden=[1 0];
PIDreg = tf(PIDgain,PIDden);         % Trasfer function of PID Controller

cltf=PIDreg*PlantS;                                  % open loop system
cltf = feedback(cltf,1,-1);          % Close loop system
[y t]= step(cltf,1:1:N);                     % take matrix [y]

error = 1-y;
if ErrorEstim==3                                          % ITAE
   error = error.*t;
end
if max(y) > 1.1                                           % eliminate overshoot
   error=error.*5;
end
error = abs(error);                                  % IAE
error = sum(error);                                  % IAE
if ErrorEstim==2                                     % ISE
   error = error.^2;
```

```
elseif ErrorEstim==3
end
fitness = 1/error;                                  % minimalise error
```

main.m

```
% main.m
%
% This script causes that the results in figure with close
% loop system  are easy to undestand by user.
% The file is called by CITfig.m
% In order to see GUI run initial window.m
%
% Author:      Daniel Czarkowski DLX4
%                            daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department%main program

try
   contr;
catch
   contr=1;    % default PID regulator
end
try
   ErrorEstim; % Type of performance index
catch                         % 1-IAE 2-ISE 3-ITAE
   ErrorEstim=1;
end

try
   Kp;
catch
   Kp=1;
   Ki=0;
   Kd=0;
end

try
   N;
catch
   N=1000;
end

try
   PlantS;
catch
   PlantS=tf(1,[180 1]);
end

[PlantNum,PlantDen] = tfdata(PlantS);                              % Get numerator and de-
nominator
```

```
[PlantNum] = deal(PlantNum{1});                                          % Transfer
from cell to array
[PlantDen] = deal(PlantDen{1});                                          % Transfer
from cell to array



for i=1:length(PlantNum)                    % eliminate zeros from array PlantNum
   if PlantNum(i)==0
      PlantNum(i)=[];
       if PlantNum(i)==0
       PlantNum(i)=[];
         if PlantNum(i)==0
            PlantNum(i)=[];
            if PlantNum(i)==0
               PlantNum(i)=[];
               if PlantNum(i)==0
               PlantNum(i)=[];
                       else break
                       end
               else break
               end
         else break
         end
      else break
      end
       else break
   end
end

for i=1:length(PlantDen)      % eliminate zeros from array PlantDen
   if PlantDen(i)==0
      PlantDen(i)=[];
       if PlantDen(i)==0
       PlantDen(i)=[];
         if PlantDen(i)==0
            PlantDen(i)=[];
            if PlantDen(i)==0
               PlantDen(i)=[];
               if PlantDen(i)==0
               PlantDen(i)=[];
                       else break
                       end
               else break
               end
         else break
         end
      else break
      end
       else break
   end
end
```

```
% Visualisation
HH = findobj(gcf,'Tag','PlantNum');
if length (PlantNum)==1
set(HH,'String',PlantNum(1));
elseif length (PlantNum)==2
   set(HH,'String',num2str([num2str(PlantNum(1)) '*s + ' num2str(PlantNum(2))]));
elseif length (PlantNum)==3
   set(HH,'String',num2str([num2str(PlantNum(1)) '*s^2 + ' num2str(PlantNum(2)), ...
               '*s + ' num2str(PlantNum(3))]));
elseif length (PlantNum)==4
   set(HH,'String',num2str([num2str(PlantNum(1)) '*s^3 + ' num2str(PlantNum(2)), ...
               '*s^2 + ' num2str(PlantNum(3)) '*s + ' num2str(PlantNum(4))]));
elseif length (PlantNum)==4
          set(HH,'String',num2str([num2str(PlantNum(1)) '*s^4 + ' num2str(PlantNum(2)),...
            '*s^3 + ' num2str(PlantNum(3)) '*s^2 + ' num2str(PlantNum(4)) ,...
            '*s + ' num2str(PlantNum(5))]));
elseif length (PlantNum)==4
                  set(HH,'String',num2str([num2str(PlantNum(1)) '*s^5 +
'num2str(PlantNum(2)),...
               '*s^4 + ' num2str(PlantNum(3)) '*s^3 + ' num2str(PlantNum(4)) ,...
               '*s^2 + ' num2str(PlantNum(5)),...
                              '*s + ' num2str(PlantNum(6))]));
else
   warning ('The plant is more than fifth order system, this program cannot accept it');
   set(HH,'String','ERROR!!!');
end



HH = findobj(gcf,'Tag','PlantDen');
if length (PlantDen)==1
set(HH,'String',PlantDen(1));
elseif length (PlantDen)==2
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s + ' num2str(PlantDen(2))]));
elseif length (PlantDen)==3
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^2 + ' num2str(PlantDen(2)),...
         '*s + ' num2str(PlantDen(3))]));
elseif length (PlantDen)==4
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^3 + ' num2str(PlantDen(2)),...
         '*s^2 + ' num2str(PlantDen(3)) '*s + ' num2str(PlantDen(4))]));
elseif length (PlantDen)==4
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^4 + ' num2str(PlantDen(2)),...
         '*s^3 + ' num2str(PlantDen(3)) '*s^2 + ' num2str(PlantDen(4)),...
         '*s + ' num2str(PlantDen(5))]));
elseif length (PlantDen)==4
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^5 + 'num2str(PlantDen(2)),...
         '*s^4 + ' num2str(PlantDen(3)) '*s^3 + ' num2str(PlantDen(4)),...
         '*s^2 + ' num2str(PlantDen(5)) '*s + ' num2str(PlantDen(6))]));
else
   warning ('The plant is more than fifth order system, this program cannot accept it');
   set(HH,'String','ERROR!!!');
end
```

```
% Visualisation of Kp Ki Kd in close loop system (file CITfig.m)
try
HH = findobj(gcf,'Tag','Kp');
set(HH,'String',num2str(['Kp=' num2str(Kp)]));
catch
HH = findobj(gcf,'Tag','Kp');
set(HH,'String','Kp=');
end

try
HH = findobj(gcf,'Tag','Ki');
set(HH,'String',num2str(['Ki=' num2str(Ki)]));
catch
HH = findobj(gcf,'Tag','Ki');
set(HH,'String','Ki=');
end

try
HH = findobj(gcf,'Tag','Kd');
set(HH,'String',num2str(['Kd=' num2str(Kd)]));
catch
HH = findobj(gcf,'Tag','Kd');
set(HH,'String','Kd=');
end
```

InitialWindowHelp.m

```
function InitialWindowHelp

% InitialWindowHelp.m
%
% The purpose of this function is to display a help
% window containing information on the function of each
% option available in the menu
%
% Author:      Daniel Czarkowski DLX4
%              daniel@czarkowski.prv.pl
%
% History:     Final Project 2000
%              Cork Institute of Technology
%              Electrical and Electronics Department


bodyTxt = [    ' The program displays 4 main zones :                                    '
' (1) Identification                                                                    '
' (2) Tuning Rule                                                                      '
' (3) Graphs                                                                           '
' (4) Close loop system and results of identification and tuning                       '
'                                                                                      '
' In addition, we have buttons such as "Grid on" "Hold on" "Clear" etc.      '
' These buttons are useful during comparison tuning rules or identifications          '
```

```
'                                                                                '
' The first of these options "Load data" has a popmenu which consist of two options:     '
' a) load datafile - loads default datafile.m which has a real data obtain from a tank   '
' b) Real time identification - this option is available only in workstation which has   '
' Data Acquisition Unit. When we choose this option we can obtain data from a tank in    '
' real time. The program tries finding solution for 15 minutes. It is possible that      '
' stops measuring when settling value is constant for 2 minutes                          '
' Next we can choose what kind of identification method we want to use. All of them are  '
' implemented for first order system. We can also enter system in fields Zeros & Poles   '
'                                                                                        '
' The second option - "tuning methods" we can decide what kind tuning method we want   '
' to use. If we want to change parameters such as population, ranges, time of counting   '
' etc we have to change it in file genhaploid.m Next option                              '
' (NCD - Nonlinear Control System Design) is available only in Matlab with optimization  '
' toolbox and NCD toolbox. This option displays a window with close loop system.
'
' Now we can initiate values entered in the main window (GUI). Next step opens NCD block  '
' set bounds and start optimization. After optimization go to GUI, click refresh button  '
' and our PID parameters will be obtained                                                 '
'                                                                                        '
' Next two options (3,4) display results                                          '

 ];

h = figure('Units','points', ...
        'Color',[0.8 0.8 0.8], ...
   'FileName','C:\Edukacja\Matlab\work\project\InitialWindowHelp.m', ...
   'Name','Cork Institute of Technology, project by Daniel Czarkowski v1.0', ...
   'NumberTitle','off', ...
        'PaperPosition',[18 180 570 420], ...
        'PaperUnits','points', ...
        'Position',[173.25 71.25 447 436.5], ...
        'Tag','Fig1', ...
        'ToolBar','none');

h = uicontrol(...
   'Style','text', ...
   'String',bodyTxt,...
   'FontSize',10,...
   'ForegroundColor',[0 0 0],...
   'HorizontalAlignment','left',...
   'BackgroundColor',[0.8706 0.8588 0.8549],...
   'Position',[10 10 570 500]);                                                  % Output body
text into Figure


h = uicontrol(...
   'Style','text', ...
   'String','Help',...
   'FontSize',24,...
   'ForegroundColor',[1 0 0],...
   'BackgroundColor',[0.8706 0.8588 0.8549],...
```

```
   'Position',[175 520 200 40]);                                                % Output
Title
```

InitialWindow.m

```
function fig = InitialWindow()

% InitialWindow.m
%
% This is main script which initialize Graphic User
% Interface. To edit this file run this script and next
% type guide in command window
%
% Author:      Daniel Czarkowski DLX4
%                           daniel@czarkowski.prv.pl
%
% History:     Final Project 2000
%              Cork Institute of Technology
%              Electrical and Electronics Department
load InitialWindow

h0 = figure('Units','points', ...
        'BackingStore','off', ...
        'Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'DoubleBuffer','on', ...
        'FileName','C:\Edukacja\Matlab\work\project\InitialWindow.m', ...
        'Name','Cork Institute of Technology, project by Daniel Czarkowski v1.0', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[79.5 97.5 580.5 408], ...
        'Resize','off', ...
        'ResizeFcn','legend(''ResizeLegend'')', ...
        'Tag','Fig1', ...
        'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','close(gcf)', ...
        'ListboxTop',0, ...
        'Position',[598 21.749063670412 160 44.32], ...
        'String','Close', ...
        'Tag','Close');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','InitialWindowHelp', ...
        'ListboxTop',0, ...
        'Position',[598 70.35955056179785 160 44.32], ...
        'String','Info', ...
        'Tag','Info');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback',mat1, ...
        'ListboxTop',0, ...
```

```
        'Position',mat2, ...
        'String','Clear', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[66.75 343.5 132 34.5], ...
        'Style','frame', ...
        'Tag','RegFrame3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[120 366 78 9], ...
        'String','Kp=', ...
        'Style','text', ...
        'Tag','Kp');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[120 344.25 78 9], ...
        'String','Kd=', ...
        'Style','text', ...
        'Tag','Kd');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[120 355 78 9], ...
        'String','Ki=', ...
        'Style','text', ...
        'Tag','Ki');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[78 366 40 9], ...
        'String','Regulator', ...
        'Style','text', ...
        'Tag','RegStaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
```

```
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[231.75 343.5 132 34.5], ...
        'Style','frame', ...
        'Tag','PlantFrame3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[252.75 352.5 108.75 6.75], ...
        'String','----------------------------------', ...
        'Style','text', ...
        'Tag','PlantFractionText');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Min',1, ...
        'Position',[252.75 356.25 108.75 9], ...
        'String','1', ...
        'Style','text', ...
        'Tag','PlantNum');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Min',1, ...
        'Position',[252.75 344.25 108.75 9], ...
        'String','180*s + 1', ...
        'Style','text', ...
        'Tag','PlantDen');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[236.25 348.75 17.25 10.5], ...
        'String','K(s)=', ...
        'Style','text', ...
        'Tag','PlantStaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[232.5 366.75 129.75 9], ...
        'String','Current Plant', ...
        'Style','text', ...
        'Tag','PlantStaticText2');
```

```
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[599 193 160 150], ...
        'Style','frame', ...
        'Tag','Frame Search', ...
        'UserData','[ ]');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[599 368 160 150], ...
        'Style','frame', ...
        'Tag','Frame Identification', ...
        'UserData','[ ]');
h1 = axes('Parent',h0, ...
        'Units','points', ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[0.501960784313725 0.501960784313725 0.501960784313725], ...
        'ColorOrder',mat3, ...
        'Position',[24 324.75 390 60], ...
        'Tag','Close loop', ...
        'XColor',[0 0 0], ...
        'XLim',[0 0.9], ...
        'XLimMode','manual', ...
        'XTickMode','manual', ...
        'YColor',[0 0 0], ...
        'YLimMode','manual', ...
        'YTickMode','manual', ...
        'ZColor',[0 0 0]);
h2 = line('Parent',h1, ...
        'Color',[0 0 0], ...
        'Interruptible','off', ...
        'Tag','ClosedLine', ...
        'XData',mat4, ...
        'YData',mat5);
h2 = line('Parent',h1, ...
        'Color',[0 0 1], ...
        'Interruptible','off', ...
        'Marker','o', ...
        'MarkerEdgeColor',[0 0 0], ...
        'MarkerFaceColor',[0 0 0], ...
        'MarkerSize',5, ...
        'Tag','ConfigurationAxesLine1', ...
        'XData',0.06, ...
        'YData',0.65);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'FontSize',16, ...
        'FontWeight','bold', ...
        'Interruptible','off', ...
        'Position',[0.04 0.54 0], ...
```

```
        'String','-', ...
        'Tag','SignText', ...
        'UserData',-1);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Interruptible','off', ...
        'Position',mat6, ...
        'Tag','ConfigurationAxesText4', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Interruptible','off', ...
        'Position',mat7, ...
        'Rotation',90, ...
        'Tag','ConfigurationAxesText3', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Interruptible','off', ...
        'Position',[-0.05722543352601155 1.379746835443038 17.32050807568877], ...
        'Tag','ConfigurationAxesText2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Interruptible','off', ...
        'Position',[0.4491329479768785 1.088607594936709 17.32050807568877], ...
        'Tag','ConfigurationAxesText1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
        'Color',[0 0 0], ...
        'Interruptible','off', ...
        'Tag','ConfigurationAxesLine2', ...
        'XData',mat8, ...
        'YData',mat9);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[600 397 50 20], ...
        'String','Zeros', ...
        'Style','text', ...
        'Tag','zero');
```

```
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat10, ...
        'ListboxTop',0, ...
        'Position',[655 397 90 20], ...
        'String','1', ...
        'Style','edit', ...
        'Tag','EditPlantNum');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat11, ...
        'ListboxTop',0, ...
        'Position',[655 377 90 20], ...
        'String','[180 1]', ...
        'Style','edit', ...
        'Tag','EditPlantDen');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[600 377 50 20], ...
        'String','Poles', ...
        'Style','text', ...
        'Tag','poles');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat12, ...
        'ListboxTop',0, ...
        'Position',[640 219 75 48], ...
        'String',mat13, ...
        'Style','popupmenu', ...
        'Tag','ErrorEstim', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[615 267 124 19], ...
        'String','Type of criterion', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[614 314 134 19], ...
        'String','Tuning method', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat14, ...
        'ListboxTop',0, ...
        'Position',[640 176 75 48], ...
        'String',mat15, ...
        'Style','popupmenu', ...
```

```
        'Tag','contr', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[615 224 124 19], ...
        'String','Type of controller', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat16, ...
        'ListboxTop',0, ...
        'Position',[617 393 114 50], ...
        'String',mat17, ...
        'Style','popupmenu', ...
        'Tag','IdentMethods', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[603 445 142 19], ...
        'String','Identification methods', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'HorizontalAlignment','right', ...
        'ListboxTop',0, ...
        'Position',[621 492 102 19], ...
        'String','Load data', ...
        'Style','text', ...
        'Tag','LoadData');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat18, ...
        'ListboxTop',0, ...
        'Position',[617 442 114 50], ...
        'String',mat19, ...
        'Style','popupmenu', ...
        'Tag','LoadData', ...
        'Value',2);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','main;', ...
        'ListboxTop',0, ...
        'Position',mat20, ...
        'String','Refresh', ...
        'Tag','Refresh');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','rtstop1', ...
        'ListboxTop',0, ...
```

```
        'Position',[617 492 45 22], ...
        'String','Stop', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback',mat21, ...
        'ListboxTop',0, ...
        'Position',[598 149.1856375183194 73.33333333333333 26.66666666666666], ...
        'String','Hold Off', ...
        'Style','togglebutton', ...
        'Tag','Hold');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback',mat22, ...
        'ListboxTop',0, ...
        'Position',[685 148.5189708516528 73 27], ...
        'String','Grid Off', ...
        'Style','togglebutton', ...
        'Tag','Grid');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat23, ...
        'ListboxTop',0, ...
        'Position',[622 266 111 48], ...
        'String',mat24, ...
        'Style','popupmenu', ...
        'Tag','SearchMethods', ...
        'Value',1);
h1 = axes('Parent',h0, ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat25, ...
        'Position',[0.03746770025839794 0.05882352941176471 0.6782945736434108
0.6856617647058824], ...
        'Tag','Axes1', ...
        'XColor',[0 0 0], ...
        'YColor',[0 0 0], ...
        'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.4980916030534351 1.018817204301075 9.160254037844386], ...
        'Tag','Axes1Text4', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.4980916030534351 -0.06451612903225823 9.160254037844386], ...
```

```
        'Tag','Axes1Text3', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-0.05534351145038167 0.4973118279569891 9.160254037844386], ...
        'Rotation',90, ...
        'Tag','Axes1Text2', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-0.05725190839694656 1.370967741935484 9.160254037844386], ...
        'Tag','Axes1Text1', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
if nargout > 0, fig = h0; end
```

Załącznik B - Callbacks

SearchMethods

```
Value=get(gcbo,'Value');
SearchMethods=Value;
if SearchMethods==1
   genhaploid;
else
ncdinit;
end
clear SearchMethods
```

Grid

```
Value=get(gcbo,'Value');
HH=findobj(gcf,'Tag','Grid');
if Value==1
gca;
grid on;
set(HH,'String',num2str('Grid On'));
else
gca;
grid off;
set(HH,'String',num2str('Grid Off'));
end;
```

Hold

```
Value=get(gcbo,'Value');
if Value==1
gca;
hold on;
HH=findobj(gcf,'Tag','Hold');
set(HH,'String',num2str('Hold On'));
else
gca;
hold off;
set(HH,'String',num2str('Hold Off'));
end;
```

Pushbutton2

```
rtstop1
```

Refresh

```
main;
```

LoadData

```
Value=get(gcbo,'Value');
LoadData=Value;
if LoadData==1
   ident_rt;
elseif  LoadData==2
try
   load datafile;
catch
disp ('cannot find the datafile.mat! please change directory where is InitialWindow.m')
end
end
try
 plot (x,Vout,'r',x,ypoly,'g');
title('Measured (red), Polynomial curve fitting (blue)');
catch
disp ('datafile is incorrect, please change directory where is InitialWindow.m');
end
clear loadData
```

IdentMethods

```
Value=get(gcbo,'Value');
IdentMethods=Value;
ident_calc;
```

Contr

```
Value=get(gcbo,'Value');
contr=Value;
```

ErrorEstim

```
Value=get(gcbo,'Value');
ErrorEstim=Value;
```

EditPlantDen & EditPlantNum

```
HH = findobj(gcf,'Tag','EditPlantDen');
EditPlantDen = str2num(get(HH,'String'));

HH = findobj(gcf,'Tag','EditPlantNum');
EditPlantNum = str2num(get(HH,'String'));

PlantS=tf(EditPlantNum,EditPlantDen);
gcf;
try
N;
Catch
N=1000;
End
```

```
sim('pid_fine_ol');
plot(yout,'k');
title('Transfer function entered by user (black)');
main;
```

Clear

```
cla;
title('');
warning off
legend('')
warning on
Kp=('');
Ki=('');
Kd=('');
main;
```

Info

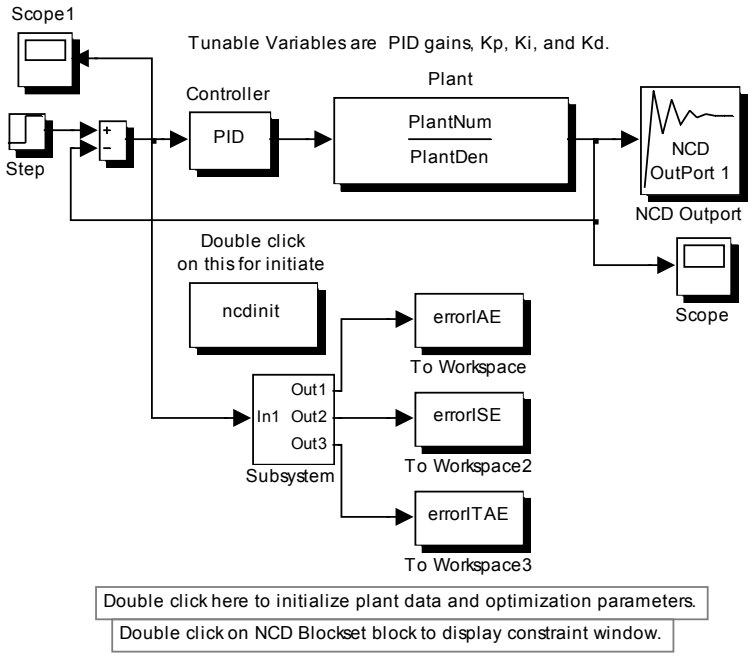
```
InitialWindowHelp
```

Close

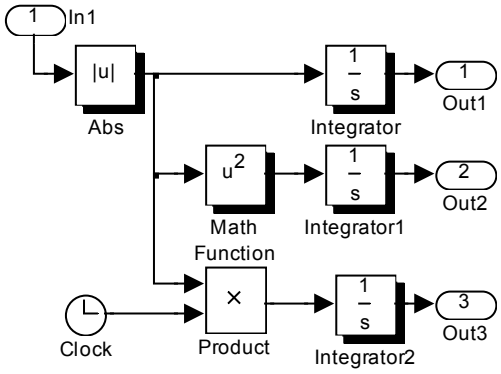
```
close(gcf)
```

Załącznik C – modele wykonane w Simulink

ncdCIT.mdl



Rys. C1 Model w Simulink, plik ncdCIT.mdl



Rys. C2 podsystem ncdCIT.mdl